Titre: Title:	Une approche tabou pour le problème d'horaires de personnel en transport aérien
Auteur: Author:	Jérôme-Olivier Ouellet
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ouellet, J.-O. (2004). Une approche tabou pour le problème d'horaires de personnel en transport aérien [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7505/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7505/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

NOTE TO USERS

UNIVERSITÉ DE MONTRÉAL

UNE APPROCHE TABOU POUR LE PROBLÈME D'HORAIRES DE PERSONNEL EN TRANSPORT AÉRIEN

JÉRÔME-OLIVIER OUELLET
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
JUIN 2004

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-97973-3
Our file Notre référence
ISBN: 0-612-97973-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

UNE APPROCHE TABOU POUR LE PROBLÈME D'HORAIRES DE PERSONNEL EN TRANSPORT AÉRIEN

présenté par : **OUELLET Jérôme-Olivier**
en vue de l'obtention du diplôme de : **Maîtrise ès sciences appliquées**
a été dûment accepté par le jury d'examen constitué de :

M. SOUMIS François, Ph.D., président
M. HERTZ Alain, Doctorat, membre et directeur de recherche
M. GAMACHE Michel, Ph.D., membre et codirecteur de recherche
M. DESAULNIERS Guy, Ph.D., membre

I'm a scientist. When I find evidence that my theories are wrong, it is as exciting as if they were correct. Scientific advance in either direction is still an advance.
Heather E. ash, New ground, Stargate Sg1

REMERCIEMENTS

J'aimerais remercier M. Alain Hertz, mon directeur de recherche, pour son support et son aide au niveau de ma recherche et de la rédaction de mon mémoire. J'ai grandement profité de ses connaissances et de sa sagesse sur le plan professionnel mais aussi sur le plan personnel. J'aimerais aussi le remercier pour son aide financière qui m'a permis de me concentrer entièrement sur mes études.

Je dois aussi remercier M. Michel Gamache, mon codirecteur, pour son aide et ses vastes connaissances en recherche opérationnelle. C'est Michel qui m'a introduit au monde de la recherche opérationnelle et qui m'a permis d'y entamer des études supérieures. Son aide à la rédaction m'a permis de voir les choses sous de nouveaux angles et ainsi de faire un travail éclairé et complet.

Un remerciement particulier doit aussi être donné à M. François Soumis avec qui j'ai débuté ma maîtrise. Son aide et ses contacts ont été grandement appréciés.

J'aimerais remercier la compagnie AD OPT Technologies ainsi que Touria El Idrissi et Olivier Milon pour le support qu'ils m'ont offert en me fournissant les données et les programmes nécessaires à la réalisation de mon travail.

J'aimerais remercier Benoît pour toute l'aide qu'il m'a donnée pendant l'élaboration de mon programme, Berthier et Michelle pour l'amour et le support qu'ils m'ont donnés toute ma vie, Jonathan, Mélanie et Alexandre pour leur support. Finalement, Sarah car elle est toujours à mes côtés dans les meilleurs et les pires moments, me donnant son amour.

RÉSUMÉ

Le projet présenté dans ce mémoire porte sur la construction d'horaires mensuels d'employés d'une importante compagnie aérienne. L'attribution de ces horaires se fait suivant un principe de séniorité stricte. Sur les coûts d'opérations, la masse salariale du personnel représente la seconde dépense en importance après les coûts de carburant, montrant ainsi l'importance d'une bonne optimisation de l'attribution de ces horaires.

Le problème se penche sur l'affectation d'horaires aux employés en utilisant le principe de ''rostering''. Les horaires des employés sont construits suivant l'ordre de séniorité entre les employés tout en tenant compte de leurs préférences. Le problème résolu s'étend sur une période mensuelle et doit traiter environ 300 employés devant effectuer environ 2000 tâches.

Le présent projet vise à implanter une méthode heuristique, basée sur la coloration de graphes, qui pourrait remplacer la méthode présentement utilisée. Le projet vise aussi à améliorer la satisfaction des employés et le temps de résolution du problème. Présentement, une heuristique appelée méthode des "compteurs" est introduite au début de la résolution du problème qui s'effectue par la technique de génération de colonnes. La méthode des "compteurs" est introduite durant les premières itérations afin d'accélérer le temps de calcul du problème maître. Les compteurs permettent de ne pas tenir compte directement des contraintes globales tout en s'assurant (de façon approximative) qu'une solution réalisable existe. Malheureusement, certaines lacunes n'ont pu être envisagées à la conception laissant entrevoir des améliorations possibles dans la résolution.

Afin d'atteindre les deux objectifs mentionnés plus haut, un nouveau programme a été élaboré ainsi que des structures propices à la nouvelle méthode. Le programme se

base sur le concept de coloration de graphes avec un algorithme tabou comme heuristique constructive. Le problème est modélisé par la coloration de graphes de façon à ce qu'une tâche soit un sommet du graphe et un employé une couleur. Suite à la modélisation, il reste seulement à trouver une coloration possible tout en tenant compte des contraintes du problème.

La nouvelle méthode est elle aussi introduite au début de la résolution du problème en remplacement de la méthode des "compteurs" avant d'utiliser la génération de colonnes. De cette façon, une fois que le problème résiduel devient compliqué à résoudre et que le temps de calcul de la nouvelle méthode augmente considérablement la résolution se poursuit par la génération de colonnes.

Suite à la création du programme, des tests ont été faits sur deux jeux de données. Chaque problème a été résolu plusieurs fois, de façon aléatoire, et des résultats moyens ont pu être amassés. En moyenne, les temps de calcul par employé sont d'environ une seconde. D'autres résultats et analyses seront donnés.

Une description des lacunes de la nouvelle méthode sera présentée ainsi que quelques avenues restant à explorer. Une description des travaux parallèles en cours ainsi que les travaux futurs sera également donnée.

ABSTRACT

The project presented in this master's thesis focuses on the monthly construction of crew schedules for important airline companies. The attribution of those schedules follows a strict seniority principal. The wages of personnel falls second only to the cost of fuel in the overall operation costs of the company, thus showing how important the optimization of those schedules is.

The problem considers the assignment of schedules to the employees using the rostering approach where the seniority of the employees is used to determine the order in which the assignment will be done. The problem to be solved stretches across a whole month and includes around 300 employees and 2000 tasks.

This project aims to implement a heuristic method, based on graph coloring to solve the problem and that could replace the presently used solution method. This project also aims to improve the satisfaction of the employees and the computing time of the presently used method. This method called counter method is a heuristic that is introduced at the beginning of a column generation algorithm. The counter method is introduced at the beginning of the solution approach replacing temporarily the column generation algorithm in order to improve its computing time. This method utilises counters that allow the program not to consider the global constraints like task covering while making sure (approximately) that a feasible solution exists. Unfortunately, certain oversights have been made and some problems have not been foreseen leaving the door open for possible improvements of the resolution.

To meet the two objectives mentioned, a new program was developed as well as structures facilitating the computing of the problem. This program is based on the graph coloring concept using a tabu algorithm as a constructive heuristic. The problem can be

modeled as a graph coloring problem so that tasks to be done are nodes and the employees needed to do the tasks are colors. Finding a coloring of the resulting graph that satisfies all constraints is the aim of the program.

The new method is also introduced at the beginning of the resolution of the problem before utilising a column generation algorithm. This way, once the residual problem becomes to long and complicated to solve with the new method, the column generation algorithm will be used.

Tests have been done on two sets of data. Each problem has been solved several times and average statistics have been computed. In average, the computing time to solve an employee is close to one second. Other results and analysis will be given.

A description of some oversights of the new method as well as few leads needing exploration will be given. A description of parallel and future works will also be presented.

TABLE DES MATIÉRES

DEDICACE ..... **IV**

REMERCIEMENTS ..... **V**

RÉSUMÉ ..... **VI**

ABSTRACT ..... **VIII**

TABLE DES MATIÉRES ..... **X**

LISTE DES TABLEAUX ..... **XIII**

LISTE DES FIGURES ..... **XIV**

CHAPITRE 1 : INTRODUCTION ET DESCRIPTION DU PROBLÈME **1**

1.1 INTRODUCTION 1

1.2 LE CYCLE DE PLANIFICATION EN TRANSPORT AÉRIEN 1

1.3 ASSIGNATION D'HORAIRES AU PERSONNEL 2

1.4 TERMINOLOGIE ET CONVENTIONS 4

1.4.1 Terminologie *4*

1.4.2 Conventions *5*

1.4.3 Exemple de base *5*

CHAPITRE 2 : ÉVOLUTION DE LA RÉSOLUTION DU PROBLÈME **8**

2.1 LA GÉNÉRATION DE COLONNES 8

2.2 MÉTHODES HEURISTIQUES 11

2.2.1 La méthode des compteurs *11*

2.2.1.1 Tâches filles 13

2.2.1.2 Tâches mères 13

2.3 ALGORITHME DE DÉCOUPAGE 14

CHAPITRE 3 : NOUVELLE MÉTHODE DE RÉSOLUTION 16

3.1 LACUNES DES VERSIONS PRÉCÉDENTES 16

3.2 APPROCHE GÉNÉRALE 17

3.3 MODÉLISATION CHOISIE : LA COLORATION DES SOMMETS D'UN GRAPHE 18

3.4 ALGORITHME TABUCOL 20

3.4.1 Description de la méthode tabou en général 20

3.4.2 Description de Tabucol 22

3.5 LE NOUVEL ALGORITHME PROPOSÉ 22

3.5.1 Extension Tabucol 23

3.5.2 Critères définissant la recherche tabou du nouvel algorithme proposé 26

3.5.2.1 Qu'est-ce qu'une solution ? 26

3.5.2.2 Quelle est la fonction objectif que l'on doit minimiser ? 27

3.5.2.3 Comment définit-on le voisinage d'une solution ? 33

3.5.2.4 Que retrouve-t-on dans la liste tabou ? 33

3.6 ALGORITHME GÉNÉRAL 34

3.6.1 L'initialisation du problème 34

3.6.2 Recherche d'une solution initiale réalisable 35

3.6.3 Affectation des employés aux tâches 36

CHAPITRE 4 : STRUCTURES DE DONNÉES 42

4.1. STRUCTURE DE DONNÉES ORIGINALES 42

4.2. STRUCTURE POUR LE CALCUL DE LA VALEUR DES SOLUTIONS VOISINES 44

4.3. STRUCTURE D'ÉTAT 49

CHAPITRE 5 : RÉSULTATS 50

5.1 DÉTECTION D'ÉTATS CRITIQUES 50

5.2 RÉSOLUTION DU PROBLÈME 56

5.3 EFFET DES CONTRAINTES SUR LA RÉSOLUTION 60

5.4 AMÉLIORATION DES SOLUTIONS ACTUELLES 70

5.5 ACCÉLÉRATION DES CALCULS 71

CHAPITRE 6 : CONCLUSION ET TRAVAUX FUTURS **74**

BIBLIOGRAPHIE **76**

LISTE DES TABLEAUX

Tableau 1.1 : Rotations de l'exemple de base 6
Tableau 1.2 : Préaffectations de l'exemple de base 6
Tableau 1.3 : Employés de l'exemple de base 6
Tableau 4.1 : Exemple de structure ΔV1 pour le coloriage de la figure 4.1 45
Tableau 4.2 : Exemple de structure ΔV2 pour le coloriage de la figure 4.1 47
Tableau 4.3 : Exemple de structure ΔV3 pour le coloriage de la figure 4.1 48
Tableau 5.1 : Discrétisation des tâches résiduelles pour la contrainte V1 53
Tableau 5.2 : Discrétisation des tâches allemande restantes pour la contrainte V2 54
Tableau 5.3 : Données relatives à une exécution d'un problème 57
Tableau 5.4 : Temps moyens de calculs totaux 65
Tableau 5.5 : Données sur la recherche d'une solution initiale 69
Tableau 5.6 : Temps de calculs sans les principes 72

LISTE DES FIGURES

Figure 1.1 : Chevauchements des tâches dans l'exemple de base 7
Figure 2.1 : Pseudo-code décrivant la technique de génération de colonnes 10
Figure 3.1 : Discrétisation des tâches selon la méthode des compteurs 17
Figure 3.2 : Algorithme tabou de base pour un problème de minimisation 21
Figure 3.3 : Représentation du graphe à colorier 24
Figure 3.4 : Représentation du graphe sans la contrainte espagnole de la tâche 4 24
Figure 3.5 : Représentation du graphe avec la contrainte espagnole de la tâche 4 25
Figure 3.6 : Représentation en graphe du problème où les compteurs ne détectent pas de conflit 26
Figure 3.7 : Détection du conflit par la nouvelle méthode 26
Figure 3.8 : Groupes de tâches pouvant être en conflit avec une tâche t 28
Figure 3.9 : Coloriage aidant la compréhension de la fonction V1 29
Figure 3.10 : Coloriage aidant la compréhension de la fonction V2 31
Figure 3.11 : Coloriage aidant la compréhension de la fonction V3 32
Figure 3.12 : Pseudo-code représentant l'initialisation du programme 35
Figure 3.13 : Pseudo-code décrivant la partie de l'algorithme qui règle les problèmes quand la valeur de la fonction objectif n'est pas égale à zéro et qu'aucune tâche n'est en conflit 39
Figure 3.14 : Pseudo-code d'écrivant l'assignation des employés aux tâches 40
Figure 3.15 : Pseudo-code décrivant le problème en entier 41
Figure 4.1 : Coloriage utilisé pour illustrer les types de structures 44
Figure 5.1 : Temps de calcul du premier jeu de données problème original 61
Figure 5.2 : Temps de calcul du premier jeu de données sans la contrainte V2 61
Figure 5.3 : Temps de calcul du premier jeu de données sans la contrainte V3 62
Figure 5.4 : Temps de calcul du premier jeu de données sans contraintes V2 et V3 62
Figure 5.5 : Temps de calcul du deuxième jeu de données problème original 63
Figure 5.6 : Temps de calcul du deuxième jeu de données sans la contrainte V2 63
Figure 5.7 : Temps de calcul du deuxième jeu de données sans la contrainte V3 64

Figure 5.8 : Temps de calcul du deuxième jeu de données sans contraintes V2 et V3....64
Figure 5.9 : Temps de calcul du deuxième jeu modifié problème original.....66
Figure 5.10 : Temps de calcul du deuxième jeu modifié sans la contrainteV2 67
Figure 5.11 : Temps de calcul du deuxième jeu modifié sans la contrainte V3 67
Figure 5.12 : Temps de calcul du deuxième jeu modifié sans contraintes V2 et V3.....68
Figure 5.13 : Pentes des graphiques des figures 5.1 à 5.3 et 5.9 à 5.12 68

CHAPITRE 1 : INTRODUCTION ET DESCRIPTION DU PROBLÈME

1.1 Introduction

Le présent mémoire a été réalisé avec l'aide de la compagnie AD OPT Technologies, dont l'une des spécialités est l'élaboration et l'optimisation d'horaires de personnel. Les compagnies aériennes étant des clients importants pour AD OPT, un projet de recherche a pris forme pour identifier une meilleure solution pour les problèmes de confection d'horaires d'agents de bord. Ce problème est important pour les compagnies aériennes car sur l'ensemble des coûts d'opération, la masse salariale du personnel représente la seconde dépense en importance après les coûts de carburant (Gershkoff, 1989). Le problème est difficile à résoudre en raison des milliers de tâches à couvrir par des centaines d'employés et du très grand nombre de contraintes qu'il faut considérer afin de respecter les conventions collectives entre les employés et la compagnie ainsi que les règles de sécurité aérienne.

1.2 Le cycle de planification en transport aérien

Le cycle de planification en transport aérien peut être divisé en cinq étapes :

- la sélection des marchés à desservir et la confection des horaires de vol,
- le choix des avions pour desservir les destinations, et la construction des rotations d'avions,
- la construction des rotations de personnel,
- la construction des horaires mensuels,
- les modifications opérationnelles.

La sélection des destinations est réalisée par une équipe qui appuie son choix sur plusieurs critères (demande des clients, offre des compétiteurs, disponibilité des aéroports) afin de déterminer les destinations que la compagnie veut desservir. Ensuite

l'horaire (le nombre de vols et l'heure de départ du vol) est construit pour chaque destination.

Plusieurs contraintes doivent être considérées afin de déterminer le type d'avion desservant un certain vol. La capacité, la vitesse et le nombre d'avions disponibles sont quelques contraintes qui doivent être prises en compte dans les calculs. Dans le domaine du transport aérien, une rotation est une suite de vols entrecoupés de repos réglementaires qui débute et se termine à une base. La base est l'aéroport où sont basés les employés. Par exemple, la séquence de vols Montréal - New York, New York - Londres et Londres - Montréal constitue une rotation. La construction de rotations consiste à déterminer un ensemble de rotations de coût minimum qui respectent les lois gouvernementales et les conventions de travail des employés et qui couvrent tous les vols de la compagnie.

La construction des horaires mensuels consiste à établir un horaire de travail pour chaque employé qui respecte la convention collective et les règles de sécurité aérienne. Ce travail consiste, entre autre, à affecter des rotations et des repos au personnel naviguant sur un horizon à moyen terme, en général un mois.

Finalement, les modifications opérationnelles sont les changements qui peuvent être faits une fois l'horaire construit. Ce problème consiste à réparer les horaires qui ne sont plus réalisables en raison de problèmes particuliers : congé de maladie, retard à un aéroport, etc.

1.3 Assignation d'horaires au personnel

On recense trois façons principes pour affecter du personnel à des horaires dans le domaine du transport aérien. Il y a le "b*idline*", le "r*ostering*" et le "p*referential biding system*".

Le "bidline" consiste à créer d'abord des blocs mensuels impersonnels qui couvrent tous les vols. Ensuite, ces blocs sont offerts aux employés qui indiquent les blocs qu'ils souhaitent obtenir.

Les deux autres types sont similaires à quelques différences près et se basent tous les deux sur les préférences des employés. Dans ce genre d'affectation, l'horaire mensuel est fait pour chaque employé en tenant compte de ses préaffectations, mais aussi de ses préférences. Un employé peut donc indiquer ses préférences pour des vols en début de semaine, débutant tôt le matin ou pour des vols outre mers. Le "rostering" est basé sur un contexte d'équité entre les employés et est essentiellement utilisé en Europe. Un poids peut être attribué aux horaires en tenant compte des préférences des employés. La version finale des horaires sera telle que la satisfaction des employés face à leurs horaires sera plus ou moins la même. Si, pour un certain mois, un employé reçoit un meilleur ou pire horaire que les autres employés, le mois suivant son horaire sera ajusté afin d'équilibrer la satisfaction de tous les employés à long terme. Le "preferential bidding system", noté PBS, quant à lui se base sur une séniorité stricte et est utilisé en Amérique du Nord. Chaque employé soumet ses préférences. La résolution se fait en tenant compte de la séniorité des employés. L'attribution des tâches aux employés ne doit jamais se faire au détriment d'un employé plus senior ce qui implique que cet employé aura toujours l'horaire réalisable qui le satisfait le plus si une solution réalisable existe pour les autres employés. Le travail qui suit porte sur les problèmes de type Nord Américain, i.e. PBS.

Dans le cas présent, il faut donc assigner les employés aux rotations de façon à ce que toutes les rotations soient couvertes sans conflits et que chaque employé, suivant l'ordre de séniorité, ait l'horaire répondant le plus à ces préférences tout en gardant la solution globale réalisable.

1.4 Terminologie et conventions

1.4.1 Terminologie

Sachant que certains termes propres à la création d'horaires dans le domaine du transport aérien ne sont pas forcément connus de tout le monde et qu'ils sont régulièrement utilisés dans le texte qui suit, une définition de ces derniers sera donnée dans cette section.

Rotation : Suite de vols, requérant un nombre d'employés qualifiés, qui débute et se termine à la même base.

Tâche mère : Le terme sera utilisé comme un synonyme de rotation.

Tâche fille : La tâche fille est une copie d'une tâche mère. Il y a autant de tâches filles que d'employés requis pour faire la tâche mère.

Qualification : Une qualification est un attribut requis par la rotation et que certains employés seulement possèdent. Par exemple, parler espagnol est une qualification.

Repos postcourrier : Le repos post courrier est un laps de temps suivant la rotation durant lequel l'employé ne peut travailler. La durée de ce repos dépend de la durée de la rotation.

Préaffectation : Tâche déjà planifiée qui doit être réalisée par un certain employé. Celle-ci peut être un congé, un entraînement ou une tâche qui a débuté le mois précédent et qui se termine le mois courant. Ces tâches sont affectées automatiquement aux employés concernés.

Crédits de vol : Nombre d'heures de vol créditées à un employé travaillant sur une rotation ou une préaffectation, excluant les repos post courrier et temps morts entre les vols composant la rotation.

Crédits de vol minimum : Nombre de crédits de vol qu'un employé doit accumuler durant le mois.

Crédits de vol maximum : Nombre de crédits de vol qu'un employé ne peut dépasser durant le mois.

1.4.2 Conventions

Il y a certaines conventions qui ont été prises afin de faciliter la compréhension des concepts et alléger un peu le texte. Une de ces conventions est l'utilisation du mot tâche comme raccourci à tâche fille. Ce changement va aider à alléger le texte puisque les employés sont toujours assignés aux tâches filles et que la notation de tâche mère sera très peu utilisée. Aussi, il est important de préciser qu'une préaffection est une tâche qui doit être effectuée. La seule différence est qu'elle ne peut être affectée qu'à un seul employé. Elle contribuera aux crédits de vol et sera comptabilisé lors de conflit tout comme les autres tâches. Une autre convention affecte les repos postcourrier. Puisque ce repos doit obligatoirement suivre la tâche et que sa durée ne dépend que de cette dernière, la durée des rotations sera modifiée afin d'inclure le repos. Si une rotation de 20 heures doit avoir un repos de 8 heures, sa durée sera modifiée à 28 heures.

1.4.3 Exemple de base

L'exemple suivant sera utilisé tout au long du mémoire afin d'illustrer les concepts décrits dans les chapitres qui suivent. Il comprend 3 employés (Blanc (B), Gris (G), Noir (N)) qui ont certaines qualifications (tableau 1.3), 3 rotations et une préaffectation qui doit être faite par l'employé G. Le tableau 1.1 donne l'information relative à chaque rotation, incluant le nombre d'employés requis, l'heure de début et de

fin, le nombre de crédits de vol associé a chaque tâche et le nombre de qualification requises par la rotation. Le tableau 1.2 indique l'employé à qui la tâche de la préaffectation doit être assignée. Chaque employé doit cumuler entre 65 et 90 crédits de vol.

Tableau 1.1 : Rotations de l'exemple de base

Rotations	Tâches	Début	Fin	Crédits	Qualifications
1					1 x hollandais
	1	24 octobre 18h30	30 octobre 11h00	45 heures	
	2	24 octobre 18h30	30 octobre 11h00	45 heures	
2					aucunes
	3	7 octobre 7h00	17 octobre 14h00	68 heures	
3					espagnol
	4	15 octobre 22h00	25 octobre 23h00	25 heures	
4					aucunes
	5	1 octobre 21h00	8 octobre 11h00	62 heures	

Tableau 1.2 : Préaffectations de l'exemple de base

Préaffectations	Personne
5	G

Tableau 1.3 : Employés de l'exemple de base

Employés	Qualifications
N	aucune
B	Hollandais, Espagnol
G	aucune

La figure suivante illustre les chevauchements temporels existant entre les cinq tâches.

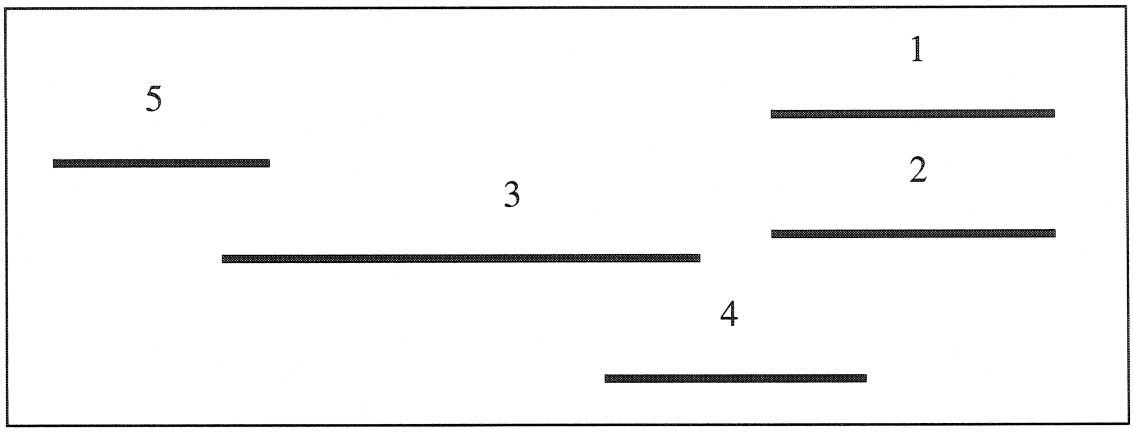


Figure 1.1 : Chevauchements des tâches dans l'exemple de base

CHAPITRE 2 : ÉVOLUTION DE LA RÉSOLUTION DU PROBLÈME

Au fil des années, plusieurs méthodes ont été implémentées pour résoudre les problèmes de confection d'horaires de personnel dans le domaine du transport aérien. Les sections qui suivent décrivent quelques-unes de ces méthodes qui ont été utilisées par la compagnie AD OPT Technologies par le passé et qui ont précédé celles présentées dans ce mémoire. Toutes les méthodes proposées s'appliquent pour le « preferential bidding » où l'optimisation se fait employé par employé selon l'ordre d'ancienneté.

2.1 La génération de colonnes

Le problème de construction d'horaires d'équipage PBS peut être formulé comme un problème de programmation linéaire en nombres entiers, plus précisément un problème de partitionnement généralisé. Pour résoudre ce problème Gamache (1995) utilise un algorithme d'évaluation et de séparation progressive. À chaque nœud de l'arbre de branchement, on résout une relaxation linéaire du problème à l'aide de la technique de génération de colonnes qui est bien adaptée pour résoudre des problèmes de très grandes tailles.

Deux types de contraintes peuvent se retrouver dans le problème. Il y a des contraintes globales qui affectent l'ensemble du problème. Par exemple, chaque employé doit avoir un horaire, chaque tâche doit être couverte par un employé et une rotation nécessite un certain nombre de personnes qualifiées. Il y a aussi des contraintes relatives à la construction des horaires des employés. Par exemple, un employé ne peut faire deux tâches en même temps et ne doit pas cumuler plus qu'un certain nombre de crédits de vol.

La technique de génération de colonnes consiste à décomposer la résolution du problème en une suite de résolutions alternées de deux types de problèmes appelés le

problème maître et les sous-problèmes. Le problème maître consiste à trouver la solution optimale au problème de partitionnement généralisé en ne tenant compte que des contraintes globales et en considérant seulement une infime partie des variables. L'algorithme du simplex est utilisé pour résoudre la relaxation linéaire du problème maître. Les sous-problèmes quant à eux se basent sur l'information donnée par le problème maître sous la forme de variables duales pour générer de nouvelles variables à insérer dans le problème maître. Dans la solution de PBS, il existe un sous-problème par employé. L'objectif des sous-problèmes est de fournir des horaires légaux qui permettent d'améliorer la solution du problème maître. Chaque sous-problème consiste à résoudre un problème de plus court chemin avec contraintes des ressources sur un graphe espace-temps propre à chaque employé. Un horaire possible pour un employé est un chemin dans son réseau. Le réseau ne contient que les tâches pouvant être effectuées par l'employé (les tâches pour lesquelles il est qualifié) et ne contient pas de tâches qui peuvent entrer en conflit avec ses préaffectations. Tant que les sous-problèmes peuvent fournir des variables au problème maître, le processus alternatif entre le problème maître et les sous-problèmes continue.

Voici une brève description de la technique de génération de colonnes avec l'algorithme d'évaluation et de séparation progressive. Premièrement, une relaxation linéaire du problème est faite au nœud initial de l'arbre avec la génération de colonnes. Une itération de la génération de colonnes commence par une génération d'horaires par les sous-problèmes. Ces horaires sont envoyés au problème maître qui trouve la solution optimale. Grâce aux variables duales, les sous-problèmes fournissent de nouveaux horaires au problème maître qui calculera la solution optimale du nouveau problème. Ceci est fait jusqu'à ce que les sous-problèmes ne puissent plus fournir des horaires pouvant améliorer la solution du problème maître. À ce moment, une décision de branchement sera prise et le tout sera réitéré jusqu'à ce que l'arbre soit exploré au complet de façon implicite et que la solution optimale entière soit trouvée.

Voici, dans la figure 2.1, un pseudo-code décrivant la technique en son entier.

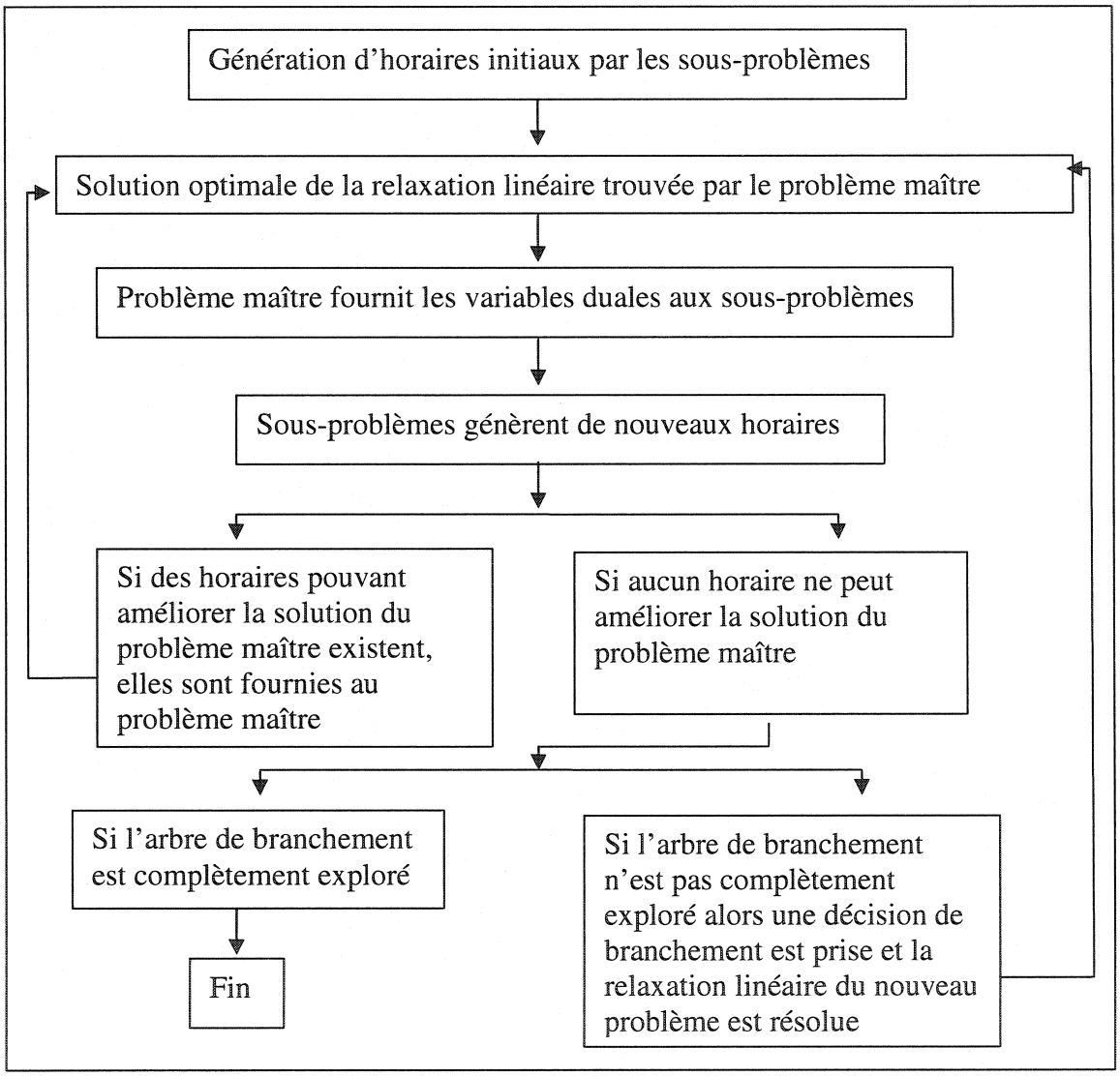


Figure 2.1 : Pseudo-code décrivant la technique de génération de colonnes

Pour ce qui est des problèmes de grande taille, la génération de colonnes peut avoir un temps de résolution très grand si elle est utilisée seule. Dans le cas de PBS avec séniorité, Gamache (1998) a utilisé l'approche suivante. Pour chaque employé du problème on résout en utilisant la démarche de la figure 2.1. Si une solution réalisable est obtenue, alors on fixe l'horaire à l'employé le plus senior et on recommence pour le

prochain employé. Sinon on retourne à l'employé précédent, on interdit son meilleur horaire et on recommence la résolution. Si on a P employés, il faut alors résoudre au moins P fois ce problème de très grande taille. La résolution d'un tel problème nécessite des temps de calculs très élevés et des heuristiques, comme la méthode des compteurs (section suivante), sont donc utilisées pour accélérer le tout. Pour ce qui est du PBS avec équité, tous les employés sont optimisés en même temps (i.e. un seul problème est résolu avec la génération de colonnes), mais ici aussi les temps de résolution sont élevés et des heuristiques sont utilisées (voir section 2.3)

2.2 Méthodes heuristiques

2.2.1 La méthode des compteurs

Vu la grande taille des problèmes à résoudre, une nouvelle approche a été conçue pour améliorer les temps de résolution des problèmes PBS avec séniorité. La résolution de la génération de colonne doit être faite pour chaque employé. Pour diminuer le temps de résolution, une heuristique est introduite durant la résolution des problèmes des premiers employés puisqu'au début de la résolution plusieurs employés sont disponibles et que la couverture des tâches est relativement facile. La méthode des compteurs traite les employés un à un, par ordre d'ancienneté. Un problème de plus court chemin est résolu pour l'employé le plus senior, et on lui assigne temporairement cet horaire. Pour ce qui est des employés résiduels, des compteurs sont mis en place afin de s'assurer qu'une solution réalisable existe satisfaisant plusieurs contraintes du problème, sans tenir compte des préférences de ces employés. Si les compteurs le permettent, l'horaire trouvé pour l'employé le plus senior lui sera donné définitivement. Sinon les compteurs permettent de savoir quels attributs sont critiques et donc quelles tâches, hors des préférences de l'employé le plus senior, ayant ces attributs devront lui être assignées ou interdites pour s'assurer de la faisabilité future du problème.

La méthode des compteurs peut donc trouver une solution pour l'employé courant en ne faisant qu'un seul calcul de plus court chemin pour trouver le meilleur horaire possible pour cet employé. En se servant des compteurs pour s'assurer que les contraintes ne sont pas violées pour les employés résiduels, les temps de calculs sont grandement diminués. D'autres compteurs sont aussi utilisés afin de déterminer quand l'utilisation des compteurs énoncés précédemment ne sera plus valide. Une fois que ces compteurs déterminent que le problème devient trop serré un retour à la génération de colonnes est fait.

Il y a donc deux types de compteurs dans la méthode. Des compteurs appelés $C1$ qui s'assurent qu'il y assez d'employés qualifiés dans le problème résiduel pour faire les tâches demandant des qualifications. Puis, il y a des compteurs appelés $C2$ qui assurent la validité de l'utilisation des compteurs $C1$. Le problème est considéré comme critique quand l'un des compteurs détecte que la demande est égale ou surpasse l'offre.

Il y a un compteur $C1$ par qualification dans le problème et par tranche de temps. Le problème fait l'objet d'une discrétisation par rapport au début et à la fin de chaque tâche. Cette discrétisation permet de voir le nombre exact de tâches se chevauchant à tout moment pour une certaine qualification. Ce compteur calcule l'offre et la demande pour une qualification donnée et assure que le problème résiduel est réalisable dans une tranche de temps donné. Une qualification devient critique quand le retrait de la personne la plus senior rend impossible la couverture des tâches résiduelles nécessitant cette qualification. Si les compteurs $C1$ détectent une qualification critique, l'employé présentement optimisé se verra imposer une tâche ayant cet attribut dans son horaire.

Les compteurs $C2$ déterminent quand l'utilisation de la méthode des compteurs doit se terminer. Les compteurs $C2$ vérifient entre autres la faisabilité globale du problème. Ils vérifient que le nombre de crédits de vol restant à assigner n'est pas trop élevé par rapport aux crédits de vol pouvant être assignés aux employés résiduels du

problème. Une fois que les compteurs $C2$ détectent que le problème devient critique, un retour à la génération de colonnes est fait.

Deux versions de cette méthode ont été créées : une qui utilise les tâches mères et une les tâches filles.

2.2.1.1 Tâches filles

Le principe étant le même pour les deux versions, seul les réseaux vont différer d'une version à l'autre. Dans la version des tâches filles (Jeandroz, 2000), chaque rotation est divisée afin d'avoir une tâche fille par employé requis dans la rotation. Les qualifications requises par les rotations sont ensuite distribuées sur les tâches filles. Le réseau global est ainsi construit de façon à permettre un flot unitaire sur les arcs. Les nœuds dans les réseaux sont donc composés de tâches filles et de tâches préaffectées aux employées. Chaque employé a son propre réseau contenant ses préaffectations ainsi que les tâches résiduelles du problème pouvant être faites par l'employé. Si un flot passe par des nœuds du réseau de l'employé, ses tâches lui sont assignées. À chaque fois qu'un horaire est assigné à un employé, le réseau de chaque employé est modifié afin d'enlever les nœuds et les arcs qui ne sont plus dans le problème.

2.2.1.2 Tâches mères

Dans la version de El Idrissi, (2002), le réseau permet un flot multiple sur les arcs, ce qui implique que chaque nœud ou tâche peut être fait par plus d'une personne. Ceci est possible puisque la création des nœuds à partir des données initiales se fait différemment que dans la version tâches filles. C'est l'affectation des qualifications des tâches qui se fait différemment. Dans la version des tâches mères, si une rotation requiert plus d'une personne ayant une certaine qualification seulement une tâche sera créée ayant cette qualification. Cependant, un flot multiple égal au nombre de personnes requis ayant la qualification est permis sur cette tâche. La taille du réseau de chaque

employé se voit ainsi grandement diminuée. Comme dans la version des tâches filles, les réseaux des employés sont modifiés à chaque fois qu'un horaire est assigné à un employé mais les arcs et les nœuds ne sont pas toujours enlevés. On diminue la valeur du flot maximal pouvant passer sur les arcs liant les nœuds des tâches assignées jusqu'à ce que le flot possible devienne zéro. C'est à ce moment que les arcs et les nœuds sont enlevés des réseaux.

2.3 Algorithme de découpage

La résolution de problèmes de grande taille (milliers d'employés et dizaine de milliers de tâches) pour le PBS avec équité ne peut se faire dans un délai raisonnable. Une approche a donc été développée (Mitchelson, 2003) afin de découper le problème en plusieurs petits morceaux plus faciles à résoudre et qui prennent donc moins de temps à résoudre.

Afin d'effectuer cette séparation, une heuristique de découpage affecte les employés et les tâches aux différents groupes de façon à ce que le profil d'offre et de demande soit semblable dans chacun des groupes.

Il devient donc évident que ce problème est compliqué car il faut tenir compte des préférences des employés et ce même si les tâches et les employés sont divisés en groupes. Une autre complication du problème est que les qualifications requises par les rotations sont distribuées sur les tâches comme pour la méthode des compteurs tâches filles. Une fois que les tâches sont mises dans des groupes, il est difficile de s'assurer que ce groupe aura le bon nombre d'employés ayant les qualifications requises. Finalement, la juxtaposition des contraintes d'offre et de demande ainsi que les qualifications causent des problèmes dans la détermination de la séparation optimale des groupes.

Pour un problème de grande taille, les résultats obtenus sont très intéressants et les temps de calculs sont beaucoup plus rapides. Vu la séparation en différents groupes, la satisfaction des employés est cependant moins bonne. Afin de respecter les préférences des employés, si le problème permet une résolution en entier, l'utilisation d'un algorithme de découpage n'est pas recommandée.

CHAPITRE 3 : NOUVELLE MÉTHODE DE RÉSOLUTION

Afin de faciliter la compréhension de la nouvelle méthode de résolution, un bref aperçu des lacunes des versions précédentes sera d'abord présenté. Suivra ensuite une description détaillée de la nouvelle approche proposée.

3.1 Lacunes des versions précédentes

La génération de colonnes est une méthode de décomposition qui permet de résoudre des programmes linéaires de très grande taille. Cependant, pour avoir une solution entière, la génération de colonnes doit être jumelée à une méthode de branchement (voir section 2.1 sur la génération de colonnes). La recherche de cette solution entière peut prendre beaucoup de temps et ceci explique en partie pourquoi l'heuristique des compteurs a été combinée avec l'approche de la génération de colonnes; on espère ainsi pouvoir accélérer le processus de résolution (voir section 2.2 sur la méthode des compteurs). La méthode des compteurs est utilisée au début de la résolution lorsque les conflits sont plus rares. Lorsque le problème devient plus restrictif (la couverture des tâches devient difficile avec le nombre d'employés restant), un retour à la génération de colonnes est effectué pour finaliser la résolution. Cette façon de faire a amélioré le temps de résolution, toutefois cette approche n'est pas parfaite. L'utilisation de l'exemple suivant permet de mettre en lumière certaines faiblesses de la méthode des compteurs. Dans l'exemple, 4 tâches doivent être assignées à 2 employés. Cependant, la tâche 1 et 2 doivent être faites par l'employé 1 car elles lui sont préaffectées. La figure 3.1 présente la discrétisation des tâches selon la méthode des compteurs.

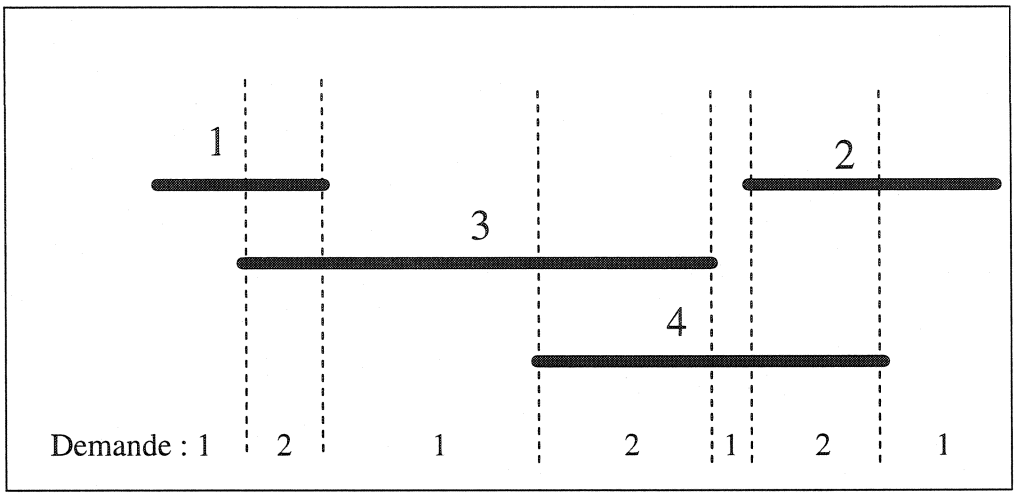


Figure 3.1 : Discrétisation des tâches selon la méthode des compteurs

Dans cet exemple, la méthode des compteurs ne détecte pas que le problème devient irréalisable. En effet, si une discrétisation du problème est faite selon le début et la fin des tâches, la demande ne dépasse jamais 2. Puisque le nombre d'employés disponible est de 2, l'offre est toujours plus grande ou égale à la demande. La méthode des compteurs ne détecte pas que les tâches 3 et 4 se chevauchent et ne peuvent être assignées à l'employé 2. De plus, elles ne peuvent pas être assignées à l'employé 1 car elles chevauchent les préaffectations de l'employé 1.

Dans les prochaines sections, nous décrirons l'approche générale ainsi que la nouvelle méthode proposée et nous tenterons de montrer qu'elle est en mesure de détecter le problème de non-faisabilité dans l'exemple décrit ci-dessus tout en ayant un temps de résolution adéquat.

3.2 Approche générale

Afin de mieux comprendre le choix de la méthode proposée dans les sections suivantes, voici un petit résumé du problème à résoudre.

Le problème consiste à trouver des horaires aux employés qui couvrent toutes les tâches à faire durant le mois. Parmi les contraintes à respecter, il faut s'assurer que les employés affectés aux tâches satisfassent les exigences relatives aux qualifications requises par ces tâches, qu'ils cumulent un nombre de crédits de vol compris entre deux bornes et qu'ils n'effectuent pas deux tâches simultanément. L'affectation des employés aux tâches doit se faire selon la séniorité des employés.

Séquentiellement, du plus senior au moins senior, l'employé se verra affecter son meilleur horaire, basé sur les préférences indiquées par ce dernier au début du mois. Si une telle affectation rend la résolution du problème résiduel impossible, l'employé verra son horaire modifié afin de rendre le problème résiduel réalisable.

La méthode qui sera décrite plus loin est introduite afin de pallier les inconvénients de la méthode des compteurs et de détecter à quel moment certaines contraintes rendent le problème non réalisable. Cette méthode sera utilisée en remplacement de la méthode des compteurs au début de la résolution des problèmes. Pour ce qui est de déterminer quelles sont les tâches causant la non-faisabilité, un autre projet est actuellement en cours et se penche présentement sur la question (voir chapitre 6).

3.3 Modélisation choisie : La coloration des sommets d'un graphe

Dans cette section, une présentation du modèle de coloration des sommets d'un graphe sera présentée. Dans les sections suivantes, cette description permettra de bien comprendre les bases de la méthode de résolution du problème.

Un graphe G (V, A) est constitué d'un ensemble V de sommets et d'un ensemble A d'arêtes. Une arête se définit toujours comme un lien entre deux sommets de V, si la liaison entre ces deux sommets n'est pas dirigée. Le problème de la coloration des

sommets d'un graphe consiste à attribuer une couleur à chaque sommet du graphe de telle sorte que deux sommets voisins (c'est-à-dire reliés par une arête) ne reçoivent pas la même couleur. Pour un entier k, une k-coloration est définie comme étant une coloration de tous les sommets de V utilisant au plus k couleurs. Si deux sommets adjacents sont de même couleur, alors l'arête les reliant est considérée en conflit et la couleur est aussi en conflit. Pour qu'une k-coloration soit légale, toutes les arêtes doivent être sans conflit.

Il y a principalement trois types de problèmes que l'on peut être amené à résoudre lorsqu'on traite la coloration d'un graphe. Le premier consiste à minimiser directement le nombre de couleurs, en travaillant dans l'espace des colorations légales, ce nombre est le *nombre chromatique* du graphe. Le second problème consiste à fixer préalablement le nombre k de couleurs que l'on peut utiliser (k étant un nombre choisi par l'utilisateur) et à minimiser le nombre de conflits. On travaille donc dans ce cas dans l'espace des k-colorations et on vise à obtenir une k-coloration légale. Si on y parvient, il s'agit ensuite de trouver une (k-1)-coloration légale, et ainsi de suite jusqu'au moment où l'on ne parvient plus à éliminer tous les conflits. Notons qu'on peut par exemple commencer ce processus en posant $k = |V| - 1$. Enfin, le troisième problème consiste à fixer également le nombre de couleurs que l'on peut utiliser, mais les conflits sont cette fois-ci interdits. Tous les sommets ne peuvent donc pas forcément recevoir une couleur et il faut maximiser le nombre de sommets colorés.

La détermination d'une k-coloration légale étant un problème NP-dur (Karp, 1972) des heuristiques sont utilisées afin de trouver une solution de qualité raisonnable. Les algorithmes gloutons (Brélaz, 1979), les algorithmes de recuit simulés (Chams et al., 1987), les algorithmes génétiques (Fleurent et Ferland, 1996), les algorithmes de fourmis (Hertz et Costa, 1997) et les algorithmes tabou (Hertz et de Werra, 1987) sont tous des heuristiques valables pour la coloration de graphes. Une méthode particulièrement

simple et efficace est la méthode Tabucol qui est une adaptation de la méthode tabou qui sera décrite dans la prochaine section.

3.4 Algorithme Tabucol

La méthode Tabucol (Hertz et de Werra, 1987) a été choisie car elle s'applique bien à notre problème et elle a déjà fait ses preuves en coloration. Mais avant de décrire l'algorithme Tabucol, une description générale de l'algorithme tabou va être faite.

3.4.1 Description de la méthode tabou en général

La méthode tabou a été développée par Glover (1986), et indépendamment par Hansen (1986). Cette méthode est basée sur un processus itératif qui, à chaque étape, se déplace d'une solution s vers une solution s' voisine afin d'améliorer la valeur de la fonction objectif f du problème. Une solution s' est dite voisine d'une solution s si un déplacement à partir de s donne la solution s'. Toutes les solutions ne sont pas voisines de s. Certains changements ne sont pas éligibles pour diverses raisons. L'ensemble des voisins de s est noté $N(s)$. La recherche du meilleur voisin dans $N(s)$ consiste donc à trouver le meilleur déplacement à effectuer à s.

La méthode tabou consiste donc à générer une suite $s_0, s_1, s_2, \ldots,$ de solutions où s_0 est la solution initiale et où s_{i+1} est une solution voisine à s_i (i.e. $s_{i+1} \in N(s_i)$). La solution s_i est ensuite implicitement introduite dans une liste tabou. La méthode la plus couramment utilisée consiste à rendre tabou certains déplacements. Les déplacements introduits dans la liste tabou sont dits déplacements *tabou*. Cependant, le fait d'interdire les déplacements dans la liste tabou peut fortement restreindre l'ensemble des solutions voisines. Un mécanisme nommé aspiration a été introduit pour pallier ce défaut. Si un déplacement est tabou, ce mécanisme permet de lever ce statut si certaines conditions sont rencontrées. Par exemple, si un déplacement tabou permet de trouver une solution

qui est meilleure que la meilleure solution trouvée jusqu'à présent, le déplacement sera permis. La liste tabou permet de sortir d'un minimum local pour se diriger éventuellement vers une meilleure solution. Par contre, après un certain nombre d'itérations I, le déplacement sera retiré de la liste tabou et pourra être refait afin d'explorer de nouveaux chemins résultant des derniers changements effectués.

Finalement, pour décrire une heuristique tabou reliée à un problème particulier, quatre ingrédients doivent être clairement définis :

- qu'est-ce qu'une solution ?
- quelle est la fonction objectif que l'on doit minimiser ?
- comment définit-on le voisinage d'une solution ?
- que retrouve-t-on dans la liste tabou ?

Un algorithme tabou de base (problème de minimisation) est donné à la figure 3.2.

Initialisation

- trouver une solution initiale s
- poser $s^* = s$ (s^* = meilleure solution)
- $L = \varnothing$ (L = liste tabou)

Tant qu'un certain critère d'arrêt n'est pas atteint

- déterminer $s' = \text{argmin}_{s'' \in N'(s)} f(s'')$;

  où $N'(s) = \{s'' \in N(s)$ où s'' est obtenu à partir de s en effectuant un mouvement qui n'est pas dans L, sauf si un mouvement de L conduit à $f(s'') < f(s^*)\}$;

  et soit m le déplacement effectué pour obtenir s' à partir de s;
- si $s' < s^*$, poser $s^* = s'$;
- poser $s = s'$;
- mettre à jour L;
  - introduire l'inverse de m dans L;
  - si l'itération de la fin d'un déplacement tabou est atteinte, retirer ce déplacement de la liste.

Figure 3.2 : Algorithme tabou de base pour un problème de minimisation

3.4.2 Description de Tabucol

Tabucol est un algorithme tabou spécialement conçu pour résoudre des problèmes de coloration des sommets d'un graphe. Cet algorithme fixe le nombre de couleurs disponible à la coloration. Il traite le problème standard de la k-coloration dans lequel il s'agit de minimiser le nombre d'arêtes en conflit.

L'algorithme Tabucol peut également être décrit selon les quatre critères mentionnés dans la section précédente. Une solution est une *k-coloration* avec éventuellement des conflits. La fonction objectif comptabilise le nombre de conflits d'une k-coloration. Le voisinage d'une solution est créé en changeant la couleur d'un sommet à l'extrémité d'une arête en conflit. Finalement, lorsqu'on change la couleur d'un sommet x de couleur c_1 à c_2, c_1 devient tabou pour x pour un certain nombre d'itérations.

3.5 Le nouvel algorithme proposé

Il reste maintenant à définir le problème qui doit être résolu en terme de coloration des sommets d'un graphe. Chaque tâche à accomplir est un sommet et deux sommets sont reliés entre eux par une arête si leurs périodes de travail se chevauchent. Chaque employé du problème est une couleur. Si le problème consistait seulement à assigner les employés aux tâches, le problème en serait un de coloration des sommets d'un graphe d'intervalle et pourrait être résolu en temps polynomial. Mais dès qu'il y a une contrainte qui impose que deux sommets doivent avoir la même couleur, le problème devient NP-dur (Biró et al. 1992) et la résolution nécessite l'utilisation d'une heuristique.

3.5.1 Extension Tabucol

Pour résoudre le problème en utilisant la coloration des sommets d'un graphe, on ne peut utiliser Tabucol comme il a été présenté précédemment car malheureusement le problème est plus complexe. Des contraintes de qualification, des crédits de vol minimum et maximum ainsi que des règles de la convention collective et des normes de sécurité aérienne s'ajoutent aux contraintes de non-chevauchement des tâches. Introduire toutes ces contraintes et trouver une solution réalisable devient plus difficile. Voici comment ces contraintes seront considérées. Premièrement, il y a les contraintes de non-chevauchement. Cette contrainte assure que deux tâches ayant lieu en même temps ne peuvent être données au même employé. Cette contrainte est modélisée en s'assurant que deux sommets qui se chevauchent dans le temps ne reçoivent pas la même couleur. D'autre part, il y a les contraintes de qualification qui requièrent qu'un certain nombre d'employés affectés aux rotations aient certaines qualifications. Pour modéliser cette contrainte, il faudra comptabiliser parmi les sommets associés à une rotation ceux dont la couleur (i.e. l'employé) possède les qualifications requises. Finalement, il y a des contraintes sur le nombre de crédits de vol minimum et maximum qu'un employé peut accumuler. Chaque sommet contribue pour une valeur de crédits de vol et la somme de ces valeurs pour tous les sommets d'une même couleur devra se situer entre les bornes.

Pour illustrer la coloration de graphes, le problème de base sera utilisé. Le problème comporte 5 sommets et trois couleurs (Noir = Employé N, Blanc = Employé B, Gris = Employé G). Le graphe à colorier est représenté à la figure 3.3. Notez bien que le sommet 5 est déjà en gris car c'est une tâche préaffectée à l'employé G.

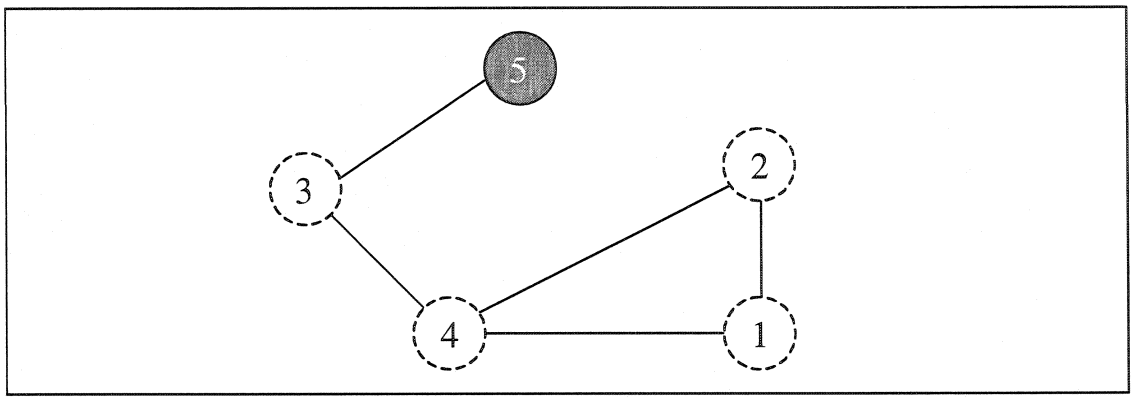


Figure 3.3 : Représentation du graphe à colorier

Si on omet le fait que la tâche 4 requiert un employé parlant l'espagnol, trois couleurs sont nécessaires pour colorier le graphe entièrement, comme l'illustre une solution possible à la figure 3.4. Les tâches 1, 2 et 4 ne peuvent pas être réalisées par le même employé puisqu'on doit éviter les chevauchements.

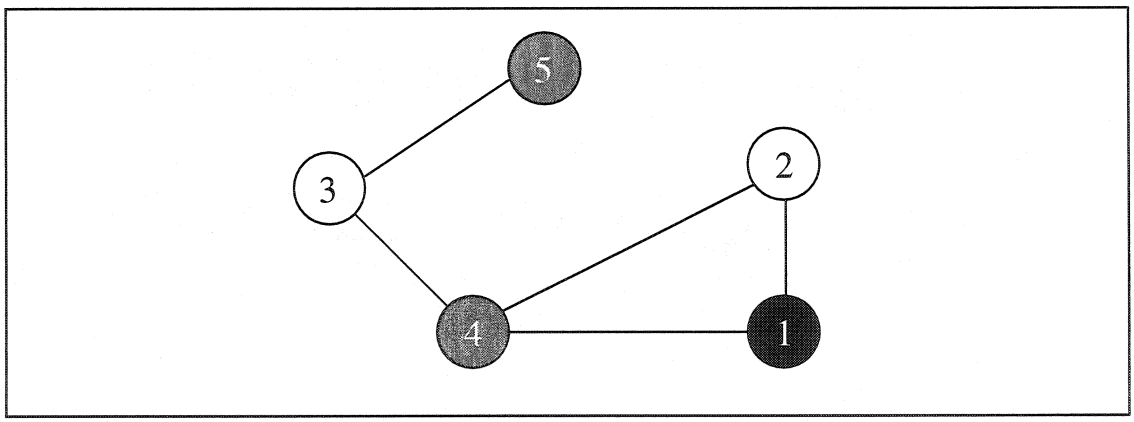


Figure 3.4 : Représentation du graphe sans la contrainte espagnole de la tâche 4

En ajoutant au problème de base la contrainte stipulant que la tâche 4 doit être attribuée à un employé parlant espagnol, il est possible de vérifier que la nouvelle méthode détecte une situation de non-faisabilité. Avec cette contrainte, le sommet 4 doit avoir la couleur blanche comme le montre la figure 3.5.

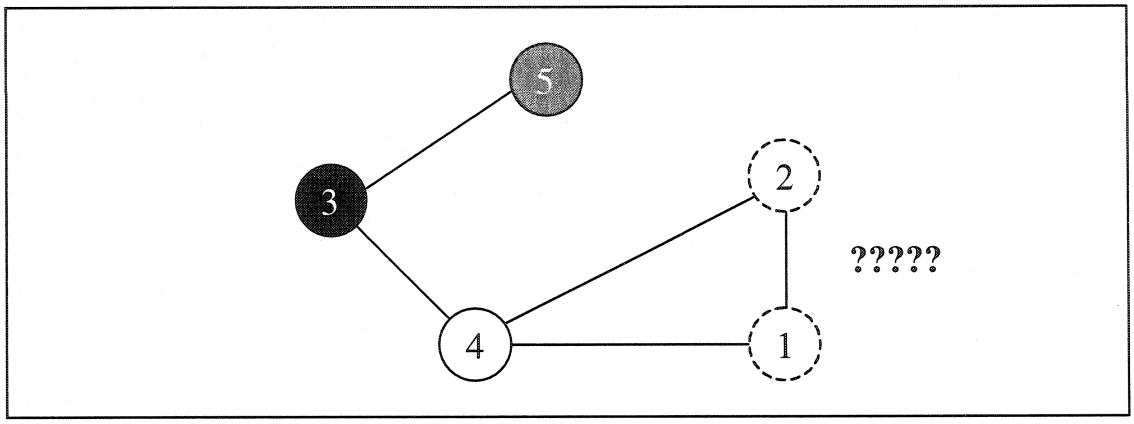


Figure 3.5 : Représentation du graphe avec la contrainte espagnole de la tâche 4

L'impossibilité de colorier le graphe avec 3 couleurs devient évidente. La tâche 1 ou 2 doit avoir la couleur blanche pour satisfaire la condition qu'au moins un employé de la rotation 1 ait la qualification hollandaise. Il est impossible de respecter cette condition puisque la tâche 4, qui est adjacente aux tâches 1 et 2, possède déjà la couleur blanche. L'illustration de la détection du problème étant faite, les sections suivantes vont montrer l'implantation de la méthode de coloriage pour la confection des horaires d'agents de bord.

La nouvelle méthode permet aussi de détecter le problème que la méthode des compteurs ne détectait pas dans la section 3.1. La représentation du problème en graphe donne la figure suivante.

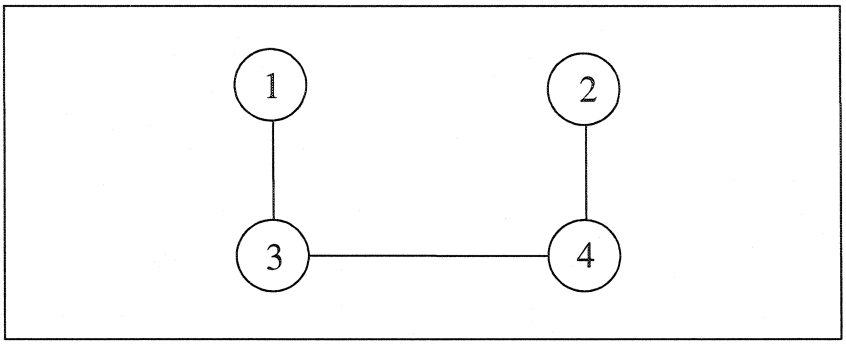


Figure 3.6 : Représentation en graphe du problème où les compteurs ne détectent pas de conflit

Si la l'employé 1(noir) doit faire les tâches 1 et 2, on peut donner la tâche 3 à l'employé 2 (gris). Il devient ensuite évident que la tâche 4 ne peut être donné ni à l'employé 1 ni à l'employé 2 (figure 3.7).

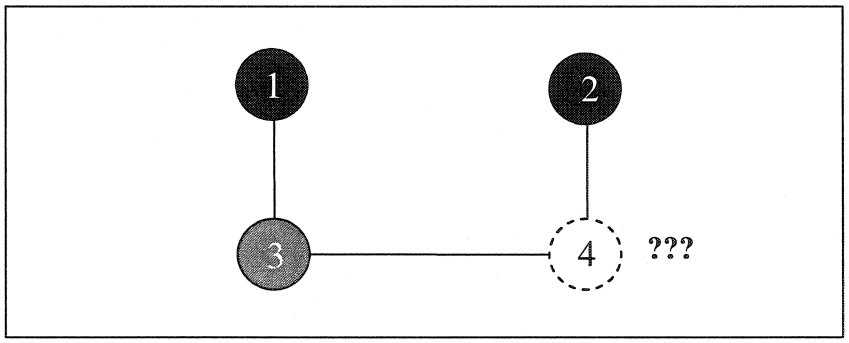


Figure 3.7 : Détection du conflit par la nouvelle méthode

3.5.2 Critères définissant la recherche tabou du nouvel algorithme proposé

Voici les quatre critères décrivant l'algorithme Tabucol modifié pour prendre en considération les nouvelles contraintes du problème étudié dans ce mémoire.

3.5.2.1 Qu'est-ce qu'une solution ?

Dans le cas présent, comme dans Tabucol, l'approche sera une variante du deuxième type de solution recherchée lorsqu'on traite un problème de coloration des

sommets d'un graphe (voir section 3.3). Chaque employé du problème aura sa propre couleur et donc le problème contient un nombre fixe de couleurs. Une solution est une coloration de tous les sommets qui contient ou non des couleurs en conflits. Le tout afin de minimiser les conflits et de trouver un coloriage légal.

3.5.2.2 Quelle est la fonction objectif que l'on doit minimiser ?

Pour une solution s, voici comment les trois types de contraintes définis dans la section précédente seront modélisés.

Les notations suivantes seront utilisées :

t = une tâche

p = un employé

q = une qualification

r = une rotation

d_t = le début de la tâche t

f_t = la fin de la tâche t

r_t = rotation associée à t

L_p = nombre de crédits de vol minimum pour l'employé p

U_p = nombre de crédits de vol maximum pour l'employé p

T = ensemble des tâches

P = ensemble des employés

Q = ensemble des qualifications

T_r = ensemble des tâches de la rotation r

$\mathsf{E}_t = \{u \in \mathsf{T} \mid (d_u \leq d_t < f_u) \text{ ou } (d_t < d_u < f_t), \text{ avec } u \neq t\}$

L'ensemble E_t est un ensemble contenant toutes les tâches qui sont en conflit avec la tâche t. Il y a deux groupes de tâches qui peuvent être en conflit avec une tâche. Soit u une des tâches en conflit avec t. Premièrement, l'heure de début de u peut être égale ou plus petite que l'heure de début de la tâche t tant que son heure de fin est plus grande que l'heure de début de t. Deuxièmement, l'heure de début de u peut être supérieure à l'heure de début de t tant que son heure de début est plus petite que l'heure de fin de t. La figure 3.6 illustre bien ces deux groupes.

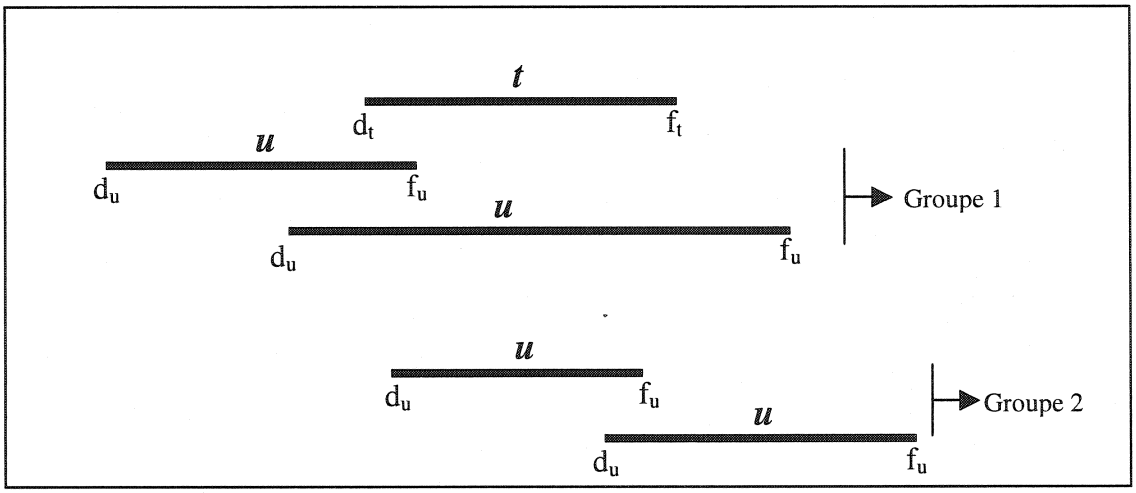


Figure 3.8 : Groupes de tâches pouvant être en conflit avec une tâche t

$p_t(s)$ = indice de l'employé affecté à la tâche t dans s

$w_p(s)$ = nombre de crédits de vol cumulés par l'employé p dans la solution s

N_{qr} = nombre d'employés requis ayant la qualification q pour la rotation r

$$Q_{pq} = \begin{cases} 1 \text{ si } p \text{ a la qualification } q \\ 0 \text{ sinon} \end{cases}$$

$$\alpha_{ut}(s) = \begin{cases} 1 \text{ si } p_u(s) = p_t(s) \\ 0 \text{ sinon} \end{cases}$$

La contrainte de chevauchement sera dite de type V1. Ce type de contraintes impose qu'un employé n'ait pas deux tâches qui se déroulent en même temps. La fonction suivante calcule le nombre de fois que cette contrainte est violée pour une certaine tâche. Pour la tâche t, attribuée à l'employé $p = p_t(s)$, la fonction calcule le nombre de tâches attribuées à p qui chevauchent en partie ou en totalité la période délimitée par la tâche t.

$$V1_t(s) = \sum_{u \in E_t} \alpha_{ut}(s)$$

La fonction précédente calcule, pour une solution s, la valeur de V1 pour la tâche t. Parmi toutes les tâches de E_t (ensemble des tâches pouvant être en conflit avec t), la fonction fait la somme de celles qui sont affectées à l'employé $p_t(s)$. La figure 3.7 donne un exemple de coloriage qui aidera à mieux comprendre cette fonction.

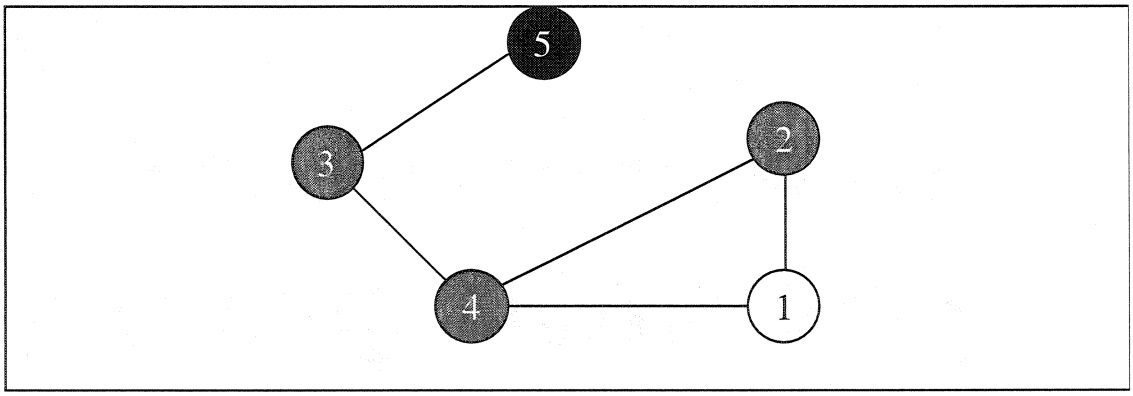


Figure 3.9 : Coloriage aidant la compréhension de la fonction V1

Dans la figure 3.7, la tâche 4, appartenant à l'employé G, a trois tâches qui peuvent être en conflit avec elle. De ces trois, deux sont faites par l'employé G, le même employé affecté à la tâche 4. On a donc $V1_4(s) = 2$ où s est la solution représentée dans la figure 3.7.

La contrainte de qualification est dite de type V2. Ce type de contrainte impose que chaque rotation contienne un nombre suffisant d'employés ayant les qualifications requises.

$$V2_t(s) = \sum_{q \in Q} (1 - Q_{p_t(s)q}) \max\left\{ N_{qr_t} - \sum_{u \in T_{r_t}} Q_{p_u(s)q}; 0 \right\}$$

La fonction ci-dessus calcule, pour une solution s, le nombre de conflits de type V2 qui sont associés à la tâche t, de la rotation r_t. Une tâche t a un conflit de type V2 si la personne $p_t(s)$ ne possède pas une des qualifications manquantes de la rotation r_t. La fonction fait la somme du nombre de conflits reliés à chaque qualification en les évaluant séparément. Le premier terme, $(1 - Q_{p_t(s)q})$, vérifie si l'employé $p_t(s)$ possède la qualification q. Si l'employé a cette qualification, l'enlever de la rotation ne pourra jamais améliorer les conflits reliés à cette qualification dans la rotation. C'est pourquoi la première soustraction donnera zéro si l'employé $p_t(s)$ possède la qualification q. Le deuxième terme $\max\left\{ N_{qr_t} - \sum_{u \in T_{r_t}} Q_{p_u(s)q}; 0 \right\}$ évalue d'abord la différence entre le nombre d'employés qualifiés exigé et le nombre d'employés qualifiés qui a été affecté à r_t. Un maximum entre la différence et zéro est utilisé car si la différence est négative (i.e. il y a un nombre supérieur d'employés ayant la qualification que d'employés requis), zéro sera choisi pour indiquer qu'aucun conflit n'existe pour cette qualification. La figure 3.8 servira à illustrer le fonctionnement de la fonction V2.

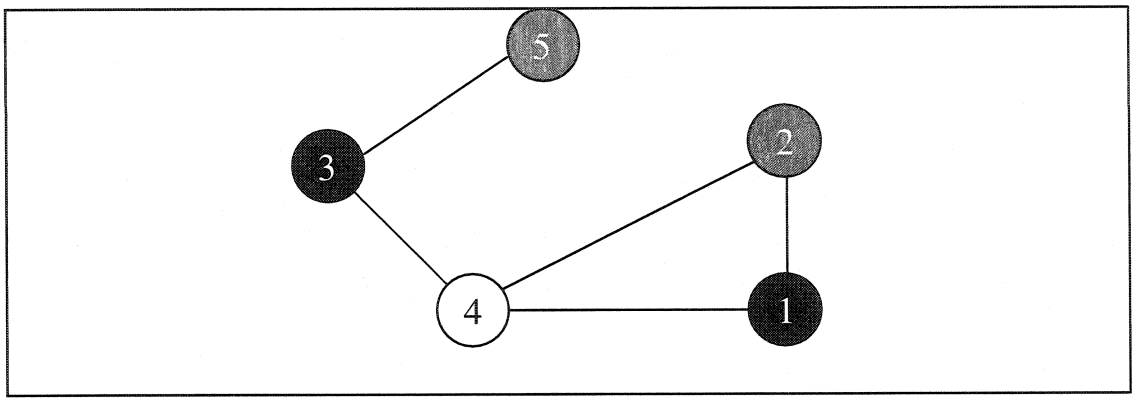


Figure 3.10 : Coloriage aidant la compréhension de la fonction V2

Par exemple, avec l'aide des tableaux 1.1 à 1.3 et de la figure 3.8, pour calculer $V2_1(s)$, i.e. le V2 de la tâche 1, il faut évaluer tous les types de qualifications requises dans la rotation 1. Seulement la qualification hollandaise est requise, donc il faut s'assurer que l'employé N possède la qualification hollandaise. Puisque l'employé N ne parle pas hollandais, la première soustraction donne 1. Pour la maximisation, la rotation 1 requiert un employé parlant hollandais et n'en possède aucun. La maximisation va donc retourner 1 et donc le calcul de la fonction V2 pour la tâche 1 donnera 1.

La contrainte de crédits de vol minimum et maximum est dite de type V3. Ce type de contraintes impose que la charge totale de travail de chaque employé soit comprise entre les bornes L_p et U_p.

$$V3_p(s) = \min\{\max\{L_p - w_p(s);0; w_p(s) - U_p\};1\}$$

La fonction ci-dessus vaut, pour une solution s, 1 si le nombre de crédits de vol accumulés par l'employé p viole les bornes de crédits de vol minimum (L_p = 65 heures) ou maximum (U_p = 90 heures). Sinon 0 est retourné. Un exemple basé sur le coloriage est donné à la figure 3.9.

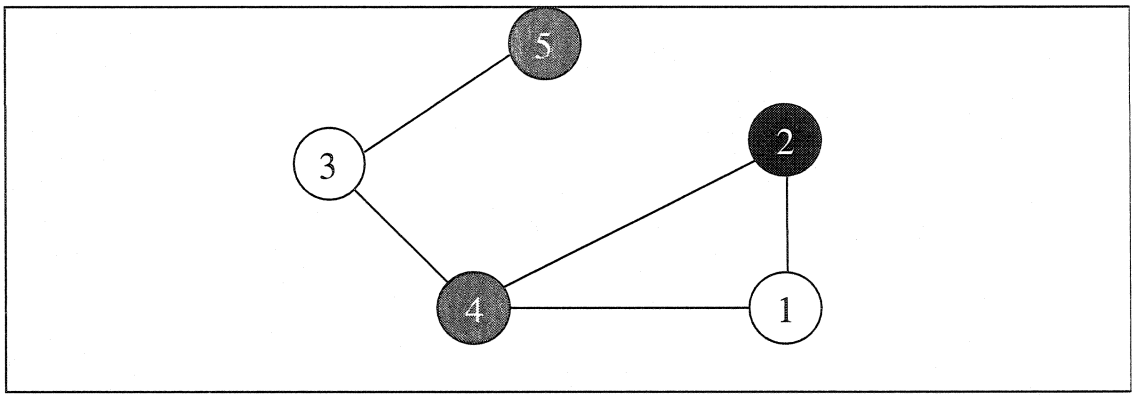


Figure 3.11 : Coloriage aidant la compréhension de la fonction V3

Avec l'aide des tableaux 1.1 à 1.3 et de la figure 3.9, on trouve que l'employé G cumule 87 heures en crédits de vol. On trouve donc $V3_G(s) = 0$ car le nombre de crédits de vol cumulés par l'employé se trouve entre les deux bornes. Pour l'employé B, il cumule 113 heures de crédits de vol et on trouve $V3_B(s) = 1$ car le nombre de crédits de vol cumulés par l'employé dépasse la borne supérieure de 23 heures. Pour ce qui est de l'employé N, il cumule 45 heures de crédits de vol et $V3_N(s) = 1$ car il lui manque 20 heures pour dépasser la borne inférieure.

Finalement, la valeur de la fonction objectif $F(s)$ utilisée dans notre algorithme tabou est obtenue en additionnant trois fonctions pondérées comme suit :

$$F(s) = \omega_1 F_1(s) + \omega_2 F_2(s) + \omega_3 F_3(s)$$

où

$$F_1(s) = \sum_{t \in T} V1_t(s)$$

$$F_2(s) = \sum_{t \in T} V2_t(s)$$

$$F_3(s) = \sum_{p \in P} V3_p(s)$$

et $\omega_1, \omega_2, \omega_3$ sont des paramètres qui permettent d'ajuster le poids de chaque terme dans la fonction objectif. Ces paramètres peuvent être modifiés en cours de résolution.

3.5.2.3 Comment définit-on le voisinage d'une solution ?

Le programme passe d'une solution à une autre en changeant la couleur d'une tâche. Certains employés ne sont pas éligibles à effectuer une tâche et dans ces cas le changement de couleur associé ne sera pas considéré pour créer une solution voisine. L'inéligibilité d'un employé à être assigné à une tâche est causée par diverses raisons : un employé peut déjà avoir une tâche de la même rotation, il peut avoir des tâches préaffectées qui sont en conflit avec la tâche ou la couleur qui le représente fait partie de la liste tabou de la tâche.

Les tâches sélectionnées pour un éventuel changement de couleur sont celles qui ont des conflits. Toutes les tâches t telles que $V1_t(s) \neq 0$ ou $V2_t(s) \neq 0$ ainsi que celles telles que $w_{p_t(s)} > U_{p_t(s)}$ sont des tâches ayant des conflits. Les tâches t telles que $w_{p_t(s)} < L_{p_t(s)}$ ne sont pas considérées comme des tâches ayant des conflits car les changements qui sont effectués dans le programme sont des changements d'employés sur les tâches. L'employé assigné à la tâche conflictuelle va perdre la tâche qui sera attribuée à un nouvel employé. Rendre une tâche conflictuelle à cause d'un manque de travail chez l'employé à qui elle appartient ne peut qu'empirer les choses. Ces tâches ayant des conflits peuvent donc faire partie d'un échange éventuel pour donner une chance à l'algorithme de diminuer la valeur actuelle de la fonction objectif.

3.5.2.4 Que retrouve-t-on dans la liste tabou ?

Une fois un changement d'employé effectué sur une tâche, l'employé qui a été enlevé de cette dernière sera mis dans la liste tabou et ne pourra pas être réaffecté à la tâche en question pendant un certain nombre d'itérations défini par l'utilisateur.

L'heuristique tabou aura aussi une fonction d'aspiration qui permettra la sélection d'un déplacement tabou seulement si la solution trouvée après ce mouvement est meilleure que la meilleure solution trouvée jusqu'à présent.

3.6 Algorithme général

La structure globale du programme permet une suite logique et rapide des opérations à exécuter afin de trouver une solution. La présente section a pour but de donner un meilleur aperçu de l'interaction entre structures et fonctions. La résolution d'un problème peut se diviser en trois parties. Il y a l'initialisation du problème, la recherche d'une solution initiale et finalement l'affectation des employés aux tâches.

3.6.1 L'initialisation du problème

Le problème est initialisé à partir des données fournies par la compagnie aérienne (introduction des rotations et des employées dans les structures de l'algorithme). Les données initiales étant maintenant toutes disponibles, un algorithme glouton est utilisé pour faire une affectation initiale des employés aux tâches. Cet algorithme prend chaque tâche du problème et évalue le résultat de son affectation à chaque employé. L'employé qui cause le moins de conflits est alors assigné à la tâche. En cas d'égalité, un choix aléatoire est effectué. La figure qui suit donne un aperçu de l'initialisation du programme.

1. Lecture des données fournies par la compagnie aérienne
2. Introduction de ces données dans les structures du programme
3. Affectation initiale à l'aide d'un algorithme glouton

    - Pour chaque tâche $t \in \mathsf{T}$

        - Pour chaque employé $p \in \mathsf{P}$

            - Calculer le nombre de conflits créés par l'affectation de la tâche t à la personne p

            - Un employé pour lequel la tâche t est la moins en conflit est assigné à cette dernière

Figure 3.12 : Pseudo-code représentant l'initialisation du programme

3.6.2 Recherche d'une solution initiale réalisable

Cette étape est une vérification de la faisabilité du problème de PBS. Avant de savoir si on peut affecter certaines tâches à certains employés en fonction de leurs préférences, il faut s'assurer que le problème résiduel a une solution qui ne contient pas de chevauchements de tâches, qui satisfait toutes les contraintes de qualification et qui fait travailler chaque employé un nombre d'heures adéquat. Cette recherche de solution initiale est une condition nécessaire à la réalisation de l'étape suivante car le reste du programme se base sur le fait qu'une telle solution existe. Cette recherche de solution ne tient pas compte des préférences des employés. Puisque l'approche utilisée est heuristique, si aucune solution initiale n'est trouvée, aucune conclusion ne peut être émise sur la non-faisabilité du problème. On peut quand même affirmer que dans un tel cas, il est très probable qu'une solution initiale rencontrant les conditions mentionnées ci-dessus n'existe pas. Pour trouver cette solution initiale, plusieurs itérations de l'algorithme présenté à la section 3.6.3 (figure 3.12) sont faites afin de trouver une solution s de valeur $F(s) = 0$ en n'enlevant aucun employé du problème.

3.6.3 Affectation des employés aux tâches

Puisque l'affectation des employés aux tâches se fait en séniorité stricte, le programme doit construire l'horaire de l'employé ayant le plus d'ancienneté. Le but est de trouver l'horaire qui satisfait le plus cet employé tout en assurant l'existence d'une solution réalisable pour les autres employés sans tenir compte de leurs préférences. L'algorithme utilisé pour le faire se base sur le fait suivant : si une solution couvrant toutes les tâches peut être trouvée en enlevant l'employé le plus senior du problème, généralement n'importe quel ensemble de tâches peut être affecté à ce dernier sans changer la faisabilité de la solution. Toutefois, retirer les tâches qui constituent le meilleur horaire de l'employé le plus senior peut causer des conflits. Il se peut que le retrait de ces tâches cause un problème de type V3. En effet, les employés dans le problème résiduel pourraient avoir un manque de travail suite au retrait des tâches. Ce manque pourrait être réglé en assignant des tâches autres que celles faisant partie du meilleur horaire de l'employé, ou bien moins de tâches devraient être données à l'employé le plus senior. Si ce problème est trouvé, il peut être un bon critère pour décider de revenir à la génération de colonnes pour terminer la résolution du problème. Quelques stratégies pour pallier ce problème seront décrites dans le chapitre 6.

Puisqu'une solution réalisable existe pour tous les employés, voir section 3.6.2, la résolution du problème associé à l'employé le plus ancien peut débuter.

On enlève l'employé le plus ancien du problème, (on colore le graphe avec une couleur en moins) et il faut donc redonner les tâches qu'il avait à d'autres. Ces tâches sont données de manière gloutonne. La recherche d'une solution s de valeur $F(s) = 0$ peut donc débuter. Pour chaque tâche ayant des conflits, choisie aléatoirement, le meilleur voisin est trouvé. Parmi ces meilleurs voisins, celui qui améliore le plus, ou empire le moins, la valeur de F, est choisi et le changement de couleur est fait. À chaque fois qu'une tâche est affectée à un nouvel employé, l'ancien employé est mis dans la liste Tabou de cette dernière pour un certain nombre d'itérations. Une fois les

ajustements faits, le processus itératif se poursuit jusqu'à ce qu'une solution $F(s) = 0$ soit trouvée ou qu'un critère d'arrêt soit atteint. Ce critère est déterminé par l'utilisateur qui peut par exemple fixer le nombre d'itérations au-delà duquel le programme doit s'arrêter faute d'avoir trouvé une solution. C'est à ce moment que l'identification des tâches causant la non faisabilité du problème est réalisée. La recherche de ces tâches va être faite par un autre programme (développé présentement par une autre étudiante) et sera décrite brièvement dans le chapitre 6. Cette recherche permet de savoir quelles tâches doivent être effectuées obligatoirement par l'employé le plus senior. Lors de la résolution du problème de l'employé précédent, une solution existait et permettait de donner les tâches voulues à cet employé tout en ayant une assignation réalisable sans conflit pour les autres employés. Pour l'employé suivant, si une solution de valeur zéro n'est pas réalisable, c'est que le retrait de cet employé rend l'assignation d'une ou plusieurs tâches impossible sans causer de conflits.

Si une solution de valeur zéro est trouvée, il faut alors donner à l'employé le plus senior les tâches composant son meilleur horaire. Puisque la version des compteurs trouvait déjà le meilleur horaire possible pour un certain employé, la recherche de cet horaire n'est pas faite par le présent programme. Tant que le problème de non faisabilité n'est pas détecté dans la résolution, l'horaire trouvé par la méthode des compteurs sera attribué à l'employé traité. Lorsqu'une solution admissible est atteinte, les tâches du meilleur horaire de l'employé le plus senior dans le problème résiduel sont enlevées et l'employé est éliminé de la liste des employées.

Remarquons qu'il est possible d'avoir une valeur de la fonction objectif différente de zéro et ce même si aucune des tâches est en conflit. Puisque les tâches appartenant à des employés ayant un manque de travail ne sont pas considérées comme étant en conflit (voir section 3.5.2), il est possible d'obtenir ce genre de situation. Dans cette éventualité, des échanges sont faits pour régler le problème, sans toutefois donner obligatoirement une valeur de zéro à la fonction objectif. Les quatre étapes suivantes

montrent comment ces échanges sont faits : 1) Un employé ayant un nombre de crédits de vol insuffisant, qui sera nommé p_i, est choisi aléatoirement. 2) Un ensemble A de q tâches pouvant être faites par l'employé p_i est sélectionné dans le problème. 3) De ces tâches, on vérifie celles qui pourraient être affectées à l'employé p_i sans causer de problème d'insuffisance pour les employés à qui ces tâches sont présentement affectées. Si aucune des tâches ne peut régler ou améliorer le problème sans causer d'autres insuffisances, on recommence à l'étape 2. Un nombre de retours r consécutifs à l'étape 2 est déterminé à l'avance. Si ce nombre est dépassé, le programme arrête car les bornes de travail des employés sont trop serrées. 4) Lorsque les tâches pouvant être données à p_i ont été trouvées, celle qui règle le problème de p_i (ou le réduit le plus) en causant le moins de conflits de type V1 et V2 est sélectionnée et les changements sont faits. Une fois les changements faits, s'il n'y a toujours pas de tâches ayant un conflit et que la fonction objectif n'est pas égale à zéro, alors on refait la procédure qui vient d'être décrite en recommençant à l'étape 1. Si des tâches ont des conflits, le programme continue son déroulement normal. La figure 3.11 illustre la partie de l'algorithme qui tient compte d'une telle éventualité.

Paramètres : entiers q et r

Tant qu'il n'existe pas de tâche en conflit alors que $F(s) \neq 0$

1) Choisir au hasard un employé p_i tel que $w_{p_i(s)} < L_{p_i}$
2) Choisir au hasard un ensemble A de q tâches
3) Si au moins une tâche de A peut être assignée à p_i sans créer de conflit de type V3 à l'employé qui effectuait cette tâche
   4) Alors assigner à p_i une telle tâche qui règle le plus de problèmes de type V3 pour p_i en causant le moins de conflit de type V1 ou V2

   Sinon
   5) Si les étapes 2) et 3) ont été effectuées r fois avec un même p_i alors STOP
   6) sinon aller à 2)

Figure 3.13 : Pseudo-code décrivant la partie de l'algorithme qui règle les problèmes quand la valeur de la fonction objectif n'est pas égale à zéro et qu'aucune tâche n'est en conflit

On passe ensuite à la résolution du prochain employé le plus senior dans la liste des employés résiduels. En se basant sur la solution trouvée pour l'employé précédent comme point de départ, on enlève les tâches qui étaient assignées à l'employé le plus senior qu'on retire du problème et recommence le processus. De cette façon, il y aura très peu de tâches ayant des conflits à arranger lors de l'optimisation du prochain employé (voir section 5.2). On continue jusqu'à ce que le problème devienne trop serré. Dans le chapitre 5, il sera montré qu'après la résolution des premiers employés, le problème devient difficile à résoudre avec le nouvel algorithme. Pour que la résolution du problème entier ne soit pas trop longue, un critère d'arrêt sera choisi et une fois ce critère atteint, le problème sera considéré trop serré et la suite de la résolution sera faite à l'aide de la génération de colonnes. Le critère choisi est un nombre maximum d'itérations, Max_iter = 30 000, après lequel la résolution est arrêtée. L'algorithme de la figure 3.12 illustre le déroulement de l'assignation des employés aux tâches.

Paramètre : entier Max_iter

1) iter = 0; $s^* = s$ (où s est la solution courante)
2) S'il existe des tâches ayant des conflits dans s alors
   - 2.1) $\hat{F} = +\infty$
   - 2.2) Pour chaque t ayant un conflit (choisie aléatoirement)
     - a) Pour chaque employé $p \neq p_t(s)$ pouvant réaliser t
       - a.1) calculer $F(s')$ où s' est la solution obtenue en assignant p à t
       - a.2) si $F(s') < F(s^*)$ alors $s = s', s^* = s, \hat{s} = s$ et aller à 2.3
       - a.3) sinon
         - a.3.1) si $F(s') < \hat{F}$ et le déplacement de s vers s' n'est pas tabou alors $\hat{F} = F(s'), \hat{s} = s'$
   - 2.3) Poser $s = \hat{s}$ et mettre à jour la liste tabou
3) Sinon, appliquer l'algorithme de la figure 3.11
4) Si iter < Max_iter
   - 4.1) alors iter = iter + 1 et aller à 2)
   - 4.2) sinon STOP

Figure 3.14 : Pseudo-code d'écrivant l'assignation des employés aux tâches

Pour terminer une description algorithmique de l'ensemble du programme est donnée à la figure 3.13.

- Appliquer l'algorithme de la figure 3.10 pour l'initialisation; soit s la solution obtenue.
- Appliquer l'algorithme de la figure 3.12 avec Max_iter = 30 000
- Si $F(s) \neq 0$ alors STOP : aucune solution admissible n'a été trouvée
- Sinon
  1) Retirer du problème l'employé p le plus senior et redistribuer ses tâches aux employés restants (de manière aléatoire)
  2) Appliquer l'algorithme de la figure 3.12 avec Max_iter = 30 000
  3) Si $F(s) \neq 0$ alors il faut imposer des tâches à p. Une fois ces tâches imposées, aller à 2)
  4) Sinon
     - 4.1) assigner à p le meilleur horaire possible selon ses préférences
     - 4.2) ôter du problème les tâches qui lui ont été assignées
     - 4.3) aller à 1)

Figure 3.15 : Pseudo-code décrivant le problème en entier

CHAPITRE 4 : STRUCTURES DE DONNÉES

Puisque les problèmes à résoudre ont environ 300 employés et 2000 tâches, il peut être nécessaire d'effectuer plusieurs milliers de changements d'affectations sur les tâches avant de trouver une solution admissible. On ne peut donc pas se permettre d'évaluer un voisin en quelques secondes puisque la résolution du problème requiert le calcul de centaines de milliers de voisins. L'accès à l'information doit aussi se faire rapidement car pour passer d'une solution à une autre, l'état conflictuel des employés et des tâches doit être connu en tout temps. C'est donc dans une optique de rapidité d'accès à l'information que les structures ont été conçues. Une description des structures de données originales, des structures pour le calcul de la valeur des solutions voisines ainsi qu'une description des structures d'état sera donné dans les sections suivantes.

4.1. Structure de données originales

Les premières structures importantes sont celles qui emmagasinent les données originales. On appelle "données originales" les données fournies par la compagnie aérienne. Deux types de données sont fournis par la compagnie : les rotations devant être faites durant le mois et les employés devant les faire. Le fichier "rotations" comprend l'information sur chaque rotation : l'heure et le jour de début, l'heure et le jour de fin, le nombre de crédits de vol, le nombre d'employés requis ainsi que les qualifications que ces employés doivent posséder. Le fichier "employés" comprend les employés, leurs qualifications, les préaffectations et leurs préférences. Trois structures, sous forme de listes, sont créées pour contenir les données originales. La première contient les tâches des rotations (nommée LT), la seconde les rotations elles-mêmes (nommée LR) et la dernière les employés (nommée LE).

Voici ce que les trois listes originales contiennent. La liste LT est composée d'objets "tâche". Un objet "tâche" comprend le jour et l'heure de début, le jour et l'heure de fin, le nombre de crédits de vol, une liste des employés qui sont tabou pour la tâche. L'objet "tâche" contient aussi une liste spéciale appelée liste d'employés interdits. Si une tâche est en conflit avec une préaffectation d'un employé, cet employé sera mis dans la liste des employés interdits pour la tâche. Cette dernière liste est très importante car elle diminue de beaucoup la liste des voisins possibles pour une tâche. Il faut noter la différence entre la liste des employés interdits et la liste tabou. La liste des employés interdits contient les employés qui ne pourront jamais être assignés à la tâche tandis que la liste tabou contient les employés ne pouvant pas présentement (et pour un nombre défini d'itérations) faire la tâche.

La liste LE se compose d'objets "employé". Un objet "employé" contient ses qualifications, le nombre d'heures travaillées, les tâches qui lui sont affectées ainsi que ses préaffectations.

Finalement, la liste LR est composée d'objets "rotation". L'objet "rotation" contient le nombre d'employés requis, le nombre et le type de qualifications requises.

Il y a beaucoup d'autres listes dans le programme mais elles sont composées de références aux éléments des trois premières listes suivant le principe énoncé précédemment.

4.2. Structure pour le calcul de la valeur des solutions voisines

Le deuxième type de structure est relié au calcul de la valeur des solutions voisines. Ce calcul est l'étape la plus effectuée durant la résolution du problème (voir ligne 2.2.a.1 dans la figure 3.12). Dans une itération de la méthode tabou, on évalue, pour chaque tâche ayant des conflits, quelle est la variation de la fonction objectif pour chaque solution voisine possible. Ce calcul évalue la variation des trois fonctions V1, V2 et V3 pour chaque déplacement vers une solution voisine, tel que mentionné dans la section 3.5. Trois structures (ΔV1, ΔV2 et ΔV3) ont été créées pour représenter ces variations. Les trois structures sont des tableaux de dimension |T| x |P|, où |T| représente le nombre de tâches, sans compter les préaffectations, et |P| le nombre d'employés dans le problème. Les valeurs contenues dans les structures diffèrent d'un type de structure à l'autre. Pour illustrer chaque type de structure nous allons utiliser l'exemple de base coloré de la façon suivante.

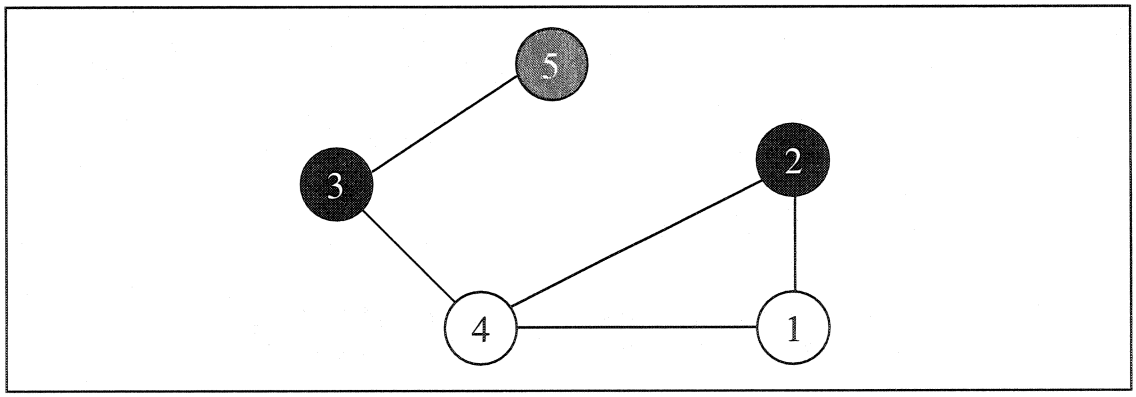


Figure 4.1 : Coloriage utilisé pour illustrer les types de structures

Structure ΔV1

La fonction V1 évalue les conflits de chevauchement entre les tâches. Quand un changement doit être fait, un employé différent doit être assigné à une tâche. Cette nouvelle assignation implique un possible changement de la valeur du V1 dans la fonction objectif en passant d'une solution à une autre. Un tel changement peut créer de

nouveaux conflits et peut en régler d'autres. La case (t, p) du tableau 4.1 contient la valeur $V1_t(s')$ où s' est la solution obtenue à partir de la solution courante s en affectant l'employé p à la tâche t si l'employé p ne possède pas déjà la tâche t. Dans ce cas, si $p = p_t(s)$, la case (t, p) contient la valeur $V1_t(s)$. Ainsi, si on veut calculer la variation de V1 en affectant une nouvelle personne p sur une tâche t, il suffit de faire la différence entre les valeurs inscrites dans les cases (t, p) et $(t, p_t(s))$; i.e. $V1(s') - V1(s) = V1_t(s') - V1_t(s) = \Delta V1_{(t,p)} - \Delta V1_{(t,p_t(s))}$ les cases (t, p) et $(t, p_t(s))$ du tableau ΔV1. Le tableau suivant montre la structure ΔV1 pour le coloriage du graphe de la figure 4.1.

Tableau 4.1 : Exemple de structure ΔV1 pour le coloriage de la figure 4.1

Tâches (t) \ Employés (p)	N	B	G
1	1	1	0
2	0	2	0
3	0	1	1
4	2	1	0

Par exemple, pour calculer la variation de V1 si on veut attribuer la tâche 1 à l'employé G, il suffit de calculer $\Delta V1_{(1,G)} - \Delta V1_{(1,B)} = 0 - 1 = -1$. Cette différence de -1 est due au fait que la tâche 1 était en conflit avec une autre tâche de l'employé B. En lui affectant l'employé G, ce conflit n'existe plus et la tâche n'a pas de conflit avec d'autres tâches de G. Une fois le meilleur voisin choisi, il reste à mettre à jour la structure. Dans l'exemple, on affecte l'employé G à la tâche 1 au lieu de B, il faudrait donc mettre à jour les cases des colonnes B et G. Il faut recalculer les cases de chaque tâche chez les deux employés impliqués car les tâches affectées aux deux employés ont changé. Le nombre de conflits qu'ont les tâches déjà affectées à ces employés change et le nombre de conflits qu'auraient les autres tâches chez ces employés change aussi. Pour chaque case

dans les colonnes des employés impliqués dans le changement, deux cas peuvent se produire. Premièrement, dans la case (t, p), si l'employé assigné à t dans la solution s' est p, i.e. $p_t(s') = p$, il suffit de calculer $V1_t(s') = \sum_{u \in E_t} \alpha_{ut}(s')$ le nombre de conflits qu'a la tâche t dans la nouvelle solution s'. Deuxièmement, dans la case (t, p), si l'employé assigné à t dans la solution s' n'est pas p i.e. $p_t(s') \neq p$, il faut calculer $V1_t(s'') = \sum_{u \in E_t} \alpha_{ut}(s'')$ où s'' est la solution obtenue de la solution s' en affectant à t une autre personne que $p_t(s')$ dans un changement ultérieur.

Structure ΔV2

La fonction V2 identifie les conflits reliés au manque d'employés qualifiés affectés aux rotations. En modifiant la personne affectée à une tâche, quatre cas peuvent se produire : (1) une rotation peut perdre un employé ayant une qualification dont elle a besoin, (2) elle peut gagner un employé qui comble une qualification manquante, (3) elle peut gagner un employé qui comble une qualification manquante et en perdre une autre qui en comblait déjà une ou (4) le changement peut n'avoir aucun impact. La valeur inscrite dans une case de la structure ΔV2, de coordonnée (t, p) représente le nombre de conflits créés ou réglés dans la rotation r_t en remplaçant l'employé ayant la tâche t par l'employé p. Lorsque la tâche t appartient déjà à l'employé p, la valeur de la case est donc nulle. En d'autres termes, la valeur de la case de coordonnées (t, p) est égale à $V2_t(s') - V2_t(s)$ où s' est la solution obtenue à partir de s en affectant l'employé p à la tâche t. En prenant le même exemple de coloriage que pour la structure ΔV1, on obtient la structure ΔV2 suivante.

Tableau 4.2 : Exemple de structure ΔV2 pour le coloriage de la figure 4.1

Tâches (t) \ Personnes (p)	N	B	G
1	1	0	1
2	0	0	0
3	0	0	0
4	1	0	1

À titre d'exemple, il y aurait une variation de 1 pour la fonction V2 si l'employé G était affecté à la tâche 1. Ceci est dû au fait que la rotation perd un employé parlant hollandais pour être remplacé par un qui ne le parle pas. Puisque la rotation de la tâche 1 requiert un employé parlant hollandais, un conflit est donc créé. Pour mettre la structure à jour une fois le meilleur voisin choisi, il suffit de mettre à jour les valeurs des cases de toutes les lignes associées aux tâches de la rotation r_t. Pour chaque tâche de la rotation r_t restant dans le problème, il faut calculer $V2_t(s'') - V2_t(s')$ et ce pour chaque employé p de P restant à optimiser, avec s'' la solution obtenue de la solution s' en assignant l'employé p sur la tâche t. Dans l'exemple ci-dessus, il faudrait mettre à jour les cases de chaque employé, pour toutes les tâches de la rotation r_1, soit toutes les cases des lignes un et deux.

Structure ΔV3

La fonction V3 vaut 1 si un employé a un nombre de crédits de vol au-dessus ou sous ses bornes de crédits de vol et zéro sinon. Le nouvel employé attribué à une tâche voit ses crédits de vol augmenter alors que celui qui avait la tâche les voit diminuer. La structure ΔV3 permet de déterminer rapidement si le statut d'un employé va changer suite à un échange éventuel. La case (t, p) du tableau ci-dessous contient la valeur $V3_p(s') - V3_p(s)$ où s' est la solution obtenue à partir de la solution courante s en

affectant l'employé p à la tâche t. Si $p_t(s) = p$, s' est la solution obtenue à partir de la solution courante s en enlevant l'employé p à la tâche t. Ainsi, si on veut calculer la variation de V3 en affectant un nouvel employé p sur une tâche t, il suffit de faire l'addition des valeurs inscrites dans les cases $(t, p_t(s))$ et (t, p). En effet, il faut additionner les conflits créés ou réglés de l'employé qui avait la tâche avec ceux créés ou réglés pour l'employé qui obtient la tâche. On a donc $V3(s') - V3(s) = V3_{p_t(s)}(s') - V3_{p_t(s)}(s) + V3_p(s') - V3_p(s) = \Delta V3_{(t,p_t(s))} + \Delta V3_{(t,p)}$. Le tableau suivant montre la structure ΔV3 pour le coloriage du graphe de la figure 4.1.

Tableau 4.3 : Exemple de structure ΔV3 pour le coloriage de la figure 4.1

Tâches (t) \ Personnes (p)	N	B	G
1	0	1	0
2	-1	1	0
3	0	1	0
4	0	1	-1

À titre d'exemple, si l'employé G était assigné à la tâche 1 au lieu de l'employé B, il suffirait d'additionner la case (1,B) à la case (1,G), ce qui donnerait une variation de 1. Ceci est dû au fait qu'en perdant la tâche 1, l'employé B se retrouve sous la borne minimum de 40 crédits de vol, et que l'employé G , en étant affecté à la tâche 1, règle son problème d'insuffisance de 3 heures mais se retrouve au-dessus de sa borne maximum de 17 heures. Pour mettre à jour le tableau ΔV3, il suffit de mettre à jour les colonnes des employés B et G. Dans le cas général, pour chaque tâche t du problème résiduel, il suffit de calculer $V3_p(s'') - V3_p(s')$ avec s'' la solution obtenue en affectant un nouvel employé p sur la tâche t dans s'.

4.3. Structure d'état

Il faut maintenant se pencher sur une structure très importante, soit la liste des tâches conflictuelles. Cette liste est nommée structure d'état car elle contient l'état des tâches dans la solution courante. Cette liste contient toutes les tâches considérées comme ayant des conflits (voir section 3.5.2.3). Le fait d'avoir la liste de tâches conflictuelles évite d'avoir à rechercher parmi toutes les tâches celles qui ont des conflits. Dans l'exemple de coloriage du problème de base qui a été utilisé dans la section 4.2, voici quelles sont les tâches qui seraient dans la liste des tâches conflictuelles : les tâches 2 et 3 puisqu'elles appartiennent à l'employé N dont le nombre de crédits de vol est au-dessus de la borne maximale, et les tâches 1 et 4 car elles se chevauchent et toutes deux sont affectées à l'employé B. Aucune préaffectation ne peut se retrouver dans la liste des tâches conflictuelles car elles ne peuvent changer de couleur.

Il est important de noter que, suite au retrait de l'employé le plus senior et à la redistribution de ses tâches, le nombre de tâches se trouvant dans la liste des tâches conflictuelles est petit. Seulement les tâches de l'employé le plus senior qui se trouvent maintenant en conflit chez le nouvel employé à qui elles ont été assignées, ainsi que celles avec qui elles sont en conflit, vont se retrouver dans la liste des tâches conflictuelles. On peut donc espérer que ceci aidera à diminuer les calculs nécessaires afin de trouver une solution s de valeur $F(s) = 0$.

CHAPITRE 5 : RÉSULTATS

Deux objectifs étaient visés dans la présente recherche : implanter une méthode heuristique basée sur la coloration de graphes pour résoudre un problème d'horaires de personnel dans le domaine du transport aérien ; améliorer la satisfaction des employés et le temps de résolution des méthodes présentement utilisées tout en détectant les situations où un employé doit avoir son horaire assujetti à certaines contraintes. Afin d'atteindre ces objectifs, l'algorithme décrit à la section 3.6 avec les structures de données décrites au chapitre 4 ont été implantés.

Le nouvel algorithme a été créé dans le but de remplacer la méthode des compteurs au début de la résolution du problème. La résolution devra donc débuter avec le nouvel algorithme pour être suivi par la génération de colonnes une fois que la résolution devient difficile.

Les sections suivantes décrivent les résultats obtenus à partir de données fournies par la compagnie AD OPT Technologies. Afin de pouvoir comparer nos résultats, les résultats de la méthode des compteurs vont être utilisés comme points de référence. Le premier problème utilisé contient 330 employés ainsi que 1956 tâches et le deuxième 317 employés et 1878 tâches. Le personnel ne peut travailler moins de 65 heures (3900 minutes) et pas plus de 90 (5100 minutes). En plus, il y a des contraintes de qualifications requises à bord des vols qui sont reliées aux langues parlées par les employés. Les paramètres suivants ont été utilisés durant la résolution des problèmes : I = 100 itérations, Max_Iter = 30 000 itérations, q = 150 tâches et r = 30 000 itérations.

5.1 Détection d'états critiques

Avant de parler de résultats sur la résolution des problèmes, la détection d'états critiques sera considérée, puisque la détection d'un tel état empêche la résolution du

problème d'aller plus loin. Une vérification du bon fonctionnement de cette détection sera faite et sera suivie d'une description de l'approche utilisée pour permettre au programme de résoudre le problème pour les employés suivants la détection d'états critiques.

Durant la résolution du problème, l'employé le plus senior est retiré du problème et le programme essaie de déterminer un horaire avec les employés résiduels (section 3.6.3). À un certain moment, le programme ne peut pas déterminer d'horaire réalisable sans cet employé. Pour déterminer ce moment, le critère suivant a été utilisé : si 30 000 itérations consécutives de la méthode tabou ont été faites sans trouver $F(s)=0$, la résolution est arrêtée. La valeur de 30 000 a été choisie parce qu'elle est grande mais aucune recherche n'a été faite afin de déterminer un critère d'arrêt plus pertinent. Cette étude devra avoir lieu dans les travaux futurs et sera discutée dans le chapitre 6. Si le critère d'arrêt est atteint, il y donc des contraintes qui doivent être satisfaites par l'employé retiré du problème. Cet employé devra donc être assigné à des tâches qui ne seraient peut-être pas dans ses préférences. Les contraintes devant être satisfaites seront nommées contraintes critiques. Un des objectifs premiers de cette maîtrise est de détecter les situations où des contraintes deviennent critiques. Puisque le programme ne tient pas compte des préférences des employés et qu'il ne détermine pas quelles tâches sont affectées aux employés, il est impossible d'affirmer que l'employé le plus senior, n'ayant pas encore reçu d'horaire, est forcé de faire certaines tâches en particulier lors de la détection de contraintes critiques. Il est cependant possible d'affirmer que l'employé doit faire certaines tâches pour que ces contraintes soient satisfaites. Dans le cadre de la présente recherche, le programme ne fait que détecter les situations où des contraintes deviennent critiques, il ne détermine pas quelles sont ces contraintes. Une fois un problème détecté, une recherche est faite à la main, afin de déterminer quelles sont les contraintes critiques. Pour trouver ces contraintes, les tâches résiduelles sont comparées aux employés restants.

La démarche suivante a donc été faite afin de montrer que le programme détecte un état critique au bon moment et que les tâches critiques trouvées à la main doivent vraiment être affectées à la personne la plus ancienne. Premièrement, le programme traite les employés un après l'autre jusqu'à ce qu'un état critique soit détecté. Ensuite, une recherche à la main est faite pour déterminer quelles sont les contraintes qui ont causé l'arrêt du programme. Une fois les contraintes critiques trouvées, elles sont comparées aux tâches qui vont être assignées à l'employé courant. Si des tâches satisfaisant aux contraintes critiques sont trouvées parmi ces tâches, la détection d'un état critique était donc probablement nécessaire. Les tâches sont ensuite enlevées du problème et la résolution du problème recommence. Voici un exemple qui montre la détection d'états critiques.

En utilisant le premier jeux de données, une des fois où le programme ne peut trouver de solution est quand l'employé 228 est retiré du problème. Voici l'état des contraintes du problème trouvées à la main.

Pour ce qui est des contraintes de type V1, une discrétisation des tâches résiduelles a été faite afin de savoir combien d'employés étaient nécessaires pour chaque plage de temps. Le tableau 5.1 nous montre qu'il y a deux intervalles demandant le plus d'employés par rapport à la contrainte V1. Ils se situent le 29^{e} jour du mois entre la 625^{e} minute et la 689^{e} et entre la 720^{e} minute et la 839^{e}. Dans ces deux intervalles, un total de 93 employés est requis, ce qui ne cause pas de problème car il reste 102 employés disponibles.

En ce qui concerne les contraintes de type V2, les seules qualifications pouvant être en cause sont celles que possède l'employé retiré. Le retrait de l'employé n'affecte pas les qualifications qu'il ne possède pas. Dans ce cas-ci, l'employé 228 possède deux qualifications, il parle polonais et allemand. Dans le problème résiduel, il ne reste qu'une tâche requérant un employé polonais et 22 requérant un employé allemand. La

qualification polonaise n'est pas critique dans ce cas-ci puisque deux autres employés ont la qualification et peuvent donc être assignés à la tâche. En ce qui concerne l'allemand, six employés peuvent être assignés à des tâches allemandes. Une discrétisation des tâches allemandes dans le tableau 5.2 nous montre que sept employés sont requis dans l'intervalle débutant le 14^{e} jour 1410^{e} minute et se terminant le 15^{e} jour 1214^{e} minute.

Tableau 5.1 : Discrétisation des tâches résiduelles pour la contrainte V1

Plage				Nombre d'employés requis	Plage				Nombre d'employés requis
Jour début	Heure début	Jour fin	Heure fin		Jour début	Heure début	Jour fin	Heure fin	
0	945	0	1054	4	2	1170	2	1304	48
0	1055	0	1184	13	2	1305	3	119	50
0	1185	0	1299	16	...	...	...	...	...
0	1300	0	1409	17	28	1055	28	1184	74
29	625	29	689	93			28	1409	77
							29	14	78
1	25	1	109	25	29	15	29	24	81
29	720	29	839	93			29	114	82
							29	229	85
1	230	1	624	32	29	230	29	624	90
1	625	1	719	35	29	625	29	689	93
1	720	1	1049	88	29	690	29	719	91
1	1050	1	1054	84	29	720	29	839	93
1	1055	1	1129	31	29	840	29	854	86
1	1130	1	1154	36	29	855	29	914	85
1	1155	1	1169	35	29	915	29	1049	80
1	1170	1	1179	26	29	1050	29	1054	74
1	1180	1	1424	30	29	1055	29	1094	71
1	1425	2	74	34	29	1095	29	1114	75
2	75	2	139	39	29	1115	29	1129	72
2	140	2	474	40	29	1130	29	1169	80
2	475	2	659	37	29	1170	29	1209	72
2	660	2	734	43	29	1210	29	1214	71
2	735	2	834	49	29	1215	29	1324	66
2	835	2	854	46	29	1325	29	1339	61
2	855	2	1109	48	29	1340	30	474	57
2	1110	2	1134	51	30	475	30	659	54
2	1135	2	1169	53	...	...	...	...	...

Tableau 5.2 : Discrétisation des tâches allemande restantes pour la contrainte V2

Plage				Nombre	Plage				Nombre
Jour début	Heure début	Jour fin	Heure fin	d'employés requis	Jour début	Heure début	Jour fin	Heure fin	d'employés requis
2	100	2	1304	0	17	635	17	1109	3
2	1305	4	59	1	17	1110	18	119	2
4	60	4	119	2	18	120	19	19	3
4	120	6	1039	3	19	20	19	1084	4
6	1040	6	1109	2	19	1085	20	1039	3
6	1110	7	99	1	20	1040	21	69	2
7	100	14	1410	2	15	1214	7		
7	1410	9	1234	2	21	935	22	1339	3
8	1215	9	1304	1	22	1340	23	1234	2
9	1235	10	164	2	23	1235	23	1304	1
9	1305	11	59	3	23	1305	25	59	2
10	165	11	119	4	25	60	25	119	3
11	60	12	1084	6	25	120	27	19	4
11	120	12	1394	5	27	20	27	1039	3
12	1085	14	1409	6	27	1040	28	99	2
12	1395	15	1214	7	28	100	28	1409	3
14	1410	15	1324	6	28	1410	29	1324	4
15	1215	16	994	5	29	1325	29	1339	3
15	1325	16	1234	4	29	1340	30	1234	2
16	995	16	1304	3	30	1235	32	1239	1
16	1235	17	634	4					
16	1305								

En ce qui concerne la contrainte de type V3, il faut noter que le programme ne tient pas compte d'un certain cas. Une fois une solution de $F(s) = 0$ trouvée pour un certain employé, il faut lui affecter des tâches selon ses préférences. En lui donnant ces tâches, un problème touchant les contraintes de type V3 peut se produire. En effet, le retrait des tâches qui vont être données à l'employé retiré du problème peut causer une insuffisance de crédits de vol chez certains employés du problème résiduel. Ce cas ne peut présentement pas être détecté par le programme et comme mentionné dans les sections précédentes, un moyen pouvant corriger cette lacune sera énoncé dans le chapitre 6.

Il y a cependant deux cas qui doivent être vérifiés afin de s'assurer qu'aucune autre contrainte de type V3 ne soit critique. Premièrement, si la personne retirée du problème possède des qualifications, le programme doit tenir compte de tous les autres employés ayant ces qualifications. Il doit s'assurer que ces employées puissent être affectés à ces tâches sans dépasser leur limite maximale de crédits de vol.

Deuxièmement, le programme doit aussi s'assurer que tous les employés résiduels peuvent se partager les tâches restantes sans avoir de conflits de type V3.

Pour l'employé 228, la deuxième vérification est assez simple. En additionnant les crédits de vol de toutes les tâches restantes dans le problème, un crédit de vol de 442986 minutes doit être distribué aux 102 employés résiduels. Les employés résiduels doivent donc se partager en moyennes 4343 minutes pour satisfaire les contraintes de type V3. Devant travailler entre 3900 et 5100 minutes, les employés résiduels n'auront aucun problème à se distribuer la charge de travail. Pour ce qui est de la deuxième vérification, l'employé 228 parle deux langues, le polonais et l'allemand. Pour ce qui est de la qualification polonaise, elle n'est pas critique car il y a seulement une tâche à donner à deux employés. Dans le cas de la qualification allemande, il y a 22 tâches qui nécessitent un employé parlant allemand et ces tâches peuvent être affectées aux six personnes restantes parlant allemand sans qu'elles n'aient trop de crédits de vol. Les 22 tâches totalisent 24482 crédits de vol, ce qui fait donc en moyenne 4080 pour chacun des employés ce qui se situe aussi entre les bornes.

Après vérification des trois types de contraintes, seulement la contrainte de type V2 reliée à la qualification allemande est critique dans l'intervalle donné plus haut. En vérifiant parmi les tâches qui seront données à l'employé 228 par la méthode des compteurs, il est possible de remarquer qu'une de ces tâches requiert un employé parlant allemand durant cet intervalle. Il existe deux possibilités. Soit la méthode des compteurs a détecté elle aussi qu'une telle tâche doit être forcée ou soit la tâche fait partie des préférences de l'employé et elle lui a été assignée automatiquement. Toutefois, on n'est pas en mesure de vérifier si la méthode des compteurs force des tâches en raison des mêmes contraintes ou bien si elle lui a attribué des tâches car elles faisaient parties des préférences de l'employé. Cependant, dans les essais faits sur les données, à chaque fois que le programme détecte un état critique durant la résolution, la méthode des compteurs affecte toujours à l'employé impliqué des tâches qui remplissent

les demandes des contraintes critiques. On peut tout de même supposer que ces tâches ne sont pas toutes des préférences des employés, et ceci porte à croire que le programme détecte bien les moments où certaines tâches doivent être données à l'employé le plus senior afin de terminer la résolution.

5.2 Résolution du problème

Pour juger de l'efficacité de notre algorithme, plusieurs critères sont importants. Le nombre de voisins calculés par employé ne doit pas être élevé ou du moins le temps pour les calculer ne doit pas l'être. Si trop de voisins sont considérés ou que trop de temps est mis pour calculer les gains ou pertes reliées à ces changements potentiels, l'algorithme ne pourra pas concurrencer les méthodes déjà existantes. Le nombre d'itérations de la méthode tabou requis pour chaque employé est aussi une information importante. Si les changements qui sont faits à chaque itération ne sont pas pertinents ou ne convergent pas vers une fonction objectif de coût zéro, encore une fois les temps de résolution seront trop élevés, rendant ainsi l'algorithme peu utile en pratique. Les deux jeux de données ont été résolus chacun 50 fois. Dans le tableau 5.3, une de ces résolutions concernant le premier jeu de données est présentée pour illustrer les critères mentionnés plus haut. Les données fournies dans le tableau 5.3 sont le résultat de la résolution du problème après avoir détecté plusieurs moments où des contraintes critiques existaient. Dans le premier jeu de données, le programme s'est arrêté une première fois à l'employé 164 car il y avait des contraintes critiques. Afin de les satisfaire, 4 tâches ont été retirées du problème et la résolution a recommencé de zéro. Ce phénomène s'est reproduit aux employés 195, 228, 231, 234, 240, 266, 277 et 280 pour un total de 12 autres tâches enlevées. Le problème s'est ensuite arrêté à l'employé 285, où des contraintes critiques ont encore été rencontrées. Puisque le problème commençait à montrer des signes qu'il devenait trop difficile à résoudre pour notre méthode, la résolution du problème a été arrêtée à cet endroit. C'est à ce moment qu'un retour à la génération de colonnes aurait lieu.

Tableau 5.3 : Données relatives à une exécution d'un problème

Employés	Tâches en conflit considérées	Voisins considérés	Nombre d'itérations requise	Temps de calculs
initialisation	2330	611962	667	15,07
0	1	8	1	0,64
1	2	223	1	0,59
2	1	216	1	0,6
3	1	85	1	0,74
4	3	164	2	0,74
5	0	0	0	0,58
6	16	3313	6	0,79
7	8	2043	1	0,58
8	5	1286	1	0,74
9	0	0	0	0,74
...	...	...	...	...
276	4	203	1	0,8
277	28	1092	11	0,9
278	27	1210	2	0,96
279	19	775	2	0,97
280	69	2787	20	1,91
281	7	319	1	0,67
282	11	342	6	0,79
283	31	1260	4	1,15
284	11	315	6	0,78
285	18289	771092	30000	62,09
Temps de résolution total : 292 secondes				
Meilleure solution pour l'employé 285 trouvée après 376 itérations				

Le tableau 5.3 contient l'information relative à la résolution des dix premiers employés (incluant la phase de recherche d'une solution initiale) ainsi que les dix derniers. Le tableau contient l'information requise pour juger de l'efficacité de l'algorithme. Voici une brève description de chaque colonne du tableau. La colonne des tâches en conflit considérées représente le nombre de tâches faisant partie de la liste des tâches à conflit pour lesquelles la valeur d'une ou de plusieurs solutions voisines ont été considérées. À chaque itération, la même tâche peut être considérée ce qui implique que le nombre de tâches considérées peut être plus grand que le nombre de tâches dans le problème. La colonne des voisins considérés contient le nombre de solutions voisines qui ont été calculées pour les tâches en conflit. De même, une solution voisine est un

changement de couleur sur une tâche, donc le nombre de voisins considérés dépend du nombre de tâches en conflit considérées et du nombre d'itérations et peut être plus grand que le nombre d'employés dans le problème. La colonne du nombre d'itérations requises contient le nombre d'itérations de la méthode tabou requis pour atteindre une solution $F(s)=0$ pour l'employé. Finalement, la dernière colonne contient le temps nécessaire pour faire tous les calculs reliés à l'employé. Les 50 résolutions du premier jeu de données permettent d'obtenir les données suivantes : le nombre de voisins calculés par employé (sans compter l'initialisation et l'employé 285) est bas (moyenne de 834 par employé). Ce nombre est considéré comme bas car au pire des cas on pourrait avoir un nombre égal au nombre de tâches restantes fois le nombre d'employés restant dans le problème. Le nombre d'itérations requis est aussi très bas avec une moyenne de 4.6 itérations par employé et le temps de calcul l'est encore plus avec une moyenne de 0.76 secondes pour faire toute la résolution de l'employé. Les résultats précédents se reflètent bien dans le tableau 5.3. On peut aussi noter une tendance à avoir des temps de calcul plus élevés pour les derniers employés impliquant que c'est approximativement à cet endroit que la génération de colonnes pourrait commencer. Comme mentionné à la fin du chapitre 4, le fait de donner à l'employé le plus senior les tâches de son meilleur horaire et d'assigner celles du prochain employé aux employés restants ne crée pas beaucoup de conflits chez les employés résiduels. Le nombre de tâches en conflit considérées et le nombre d'itérations de la méthode tabou sont habituellement bas. Même que pour certains employés, aucune itération de la méthode tabou n'est faite car après avoir enlevé l'employé ayant le plus de séniorité, les tâches appartenant à ce dernier peuvent être redistribuées sans causer de conflit (employés 5 et 9 dans le tableau 5.3).

Deux cas sont exceptionnels, la recherche d'une solution initiale et la recherche d'une solution pour un employé pour lequel on a détecté un problème. La recherche d'une solution initiale comprend tous les employés et toutes les tâches du problème et est faite après une assignation de départ déterminée par un algorithme glouton. La

solution de départ trouvée par l'algorithme glouton n'étant pas très bonne, beaucoup de changements et de recherche sont requis pour trouver une solution initiale s avec $F(s) = 0$. Il y a d'autres facteurs reliés aux types de contraintes qui affectent la recherche d'une solution initiale, mais ces facteurs seront décrits en détails dans la section 5.3. Pour ce qui est du dernier employé, beaucoup de voisins ont été calculés car il n'y a aucune solution pouvant mener à une solution s avec $F(s) = 0$. Comme décrit dans la section 3.6.3, un critère d'arrêt doit être choisi pour ne pas chercher cette solution indéfiniment. Dans le cas présent, le critère d'arrêt est un nombre maximum d'itérations, égal à 30 000, sans avoir trouver de solution $F(s) = 0$ (voir à la ligne 6 du tableau 3.11). Pour déterminer ce critère, aucune étude approfondie n'a été faite. Le nombre de 30 000 a été choisi car il est de beaucoup plus grand que la moyenne d'itérations nécessaires à la résolution d'un employé. Un des travaux futurs sur le projet est de définir un critère qui soit pertinent (voir chapitre 6). Si plus de 30 000 itérations sont faites pour le même employé, cet employé est considéré à problème (dans l'exemple, l'employé 285). Ceci explique donc le nombre élevé de tâches conflictuelles utilisées et le nombre élevé de voisins calculés. Dans la figure 5.3, le nombre d'itérations requis pour trouver la meilleur valeur de fonction objectif est 376. Cette donnée donne une bonne indication qu'il y a en effet un problème car pendant 29 624 itérations, aucune meilleure solution n'a pu être trouvée tandis qu'en moyenne une solution $F(s) = 0$ est trouvée en 4.6 itérations. Si un meilleur critère était utilisé, le temps de calcul pour l'employé 285 serait grandement diminué, il ne faut donc pas accorder trop d'importance présentement aux 62 secondes requises pour détecter des contraintes critiques. Finalement, le temps total de la résolution du problème est de 292 secondes. Il faut cependant soustraire un temps de 80 secondes au temps de résolution car c'est une constante qui revient durant chaque résolution. Les 80 secondes sont nécessaires pour faire l'initialisation du problème (section 3.6.1) et pour assigner aux employés les tâches trouvées par la méthode des compteurs (voir la section 3.6.3).

Le tableau 5.3 permet aussi de voir a quel point, l'utilisation de la solution de l'employé précédent comme point de départ pour le prochain employé rend son optimisation moins difficile. En prenant la solution de l'employé précédent, avec les conflits créés par l'assignation du meilleur horaire a cet employé, et en la modifiant pour redistribuer les tâches qu'avait l'employé le plus senior retiré, on pourrait avoir en théorie autant de tâches en conflit qu'il y de tâches restant dans le problème. Mais en pratique seulement 4 tâches sont en conflit en moyenne au début de l'optimisation de l'employé. Ceci permet donc à l'algorithme d'avoir des temps de calcul relativement rapide.

5.3 Effet des contraintes sur la résolution

Pour vérifier l'effet des contraintes sur la résolution du problème, le programme a été modifié. Des résolutions du problème ont été faites en enlevant des contraintes à tour de rôle. Une première série de résolutions a été faite sans le type V2 de contraintes, ensuite sans le type V3 et finalement sans les types V2 et V3. La contrainte de type V1 est toujours valide car le non-chevauchement est une contrainte de base. Cinquante résolutions ont été faites de chaque nouveau problème et ce pour les deux jeux de données originaux. Le temps qu'a pris la recherche d'un horaire pour chaque employé sera comparé. Les résolutions ont toutes été arrêtées avant l'employé 285 afin de ne pas biaiser les données. Une moyenne de ces temps de calcul est donnée dans les graphiques suivants.

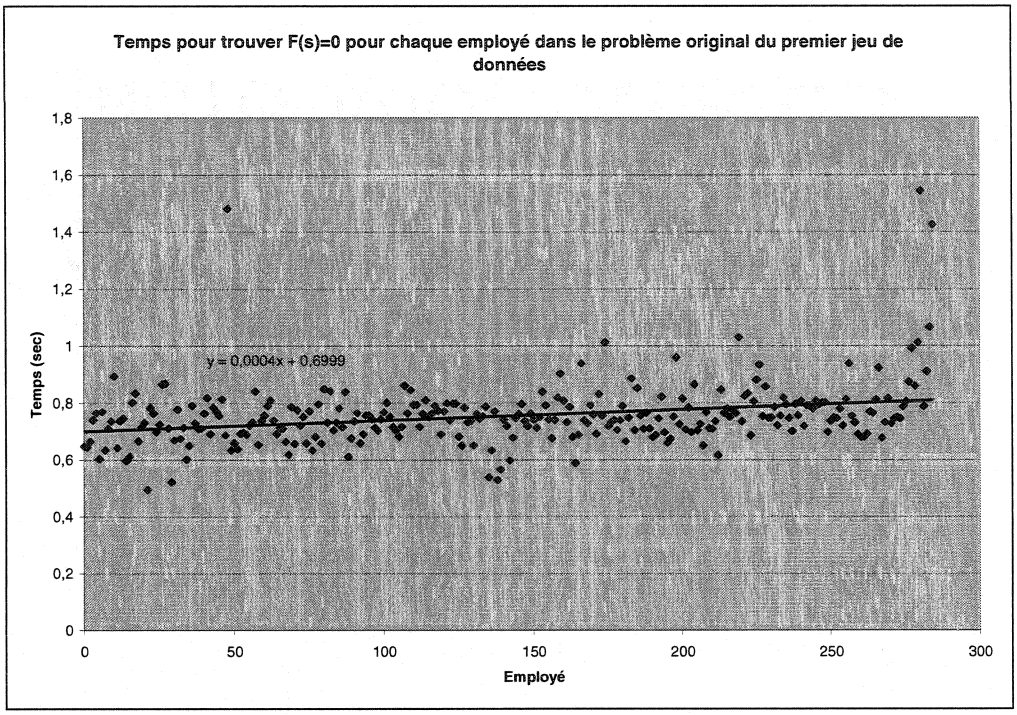


Figure 5.1 : Temps de calcul du premier jeu de données problème original

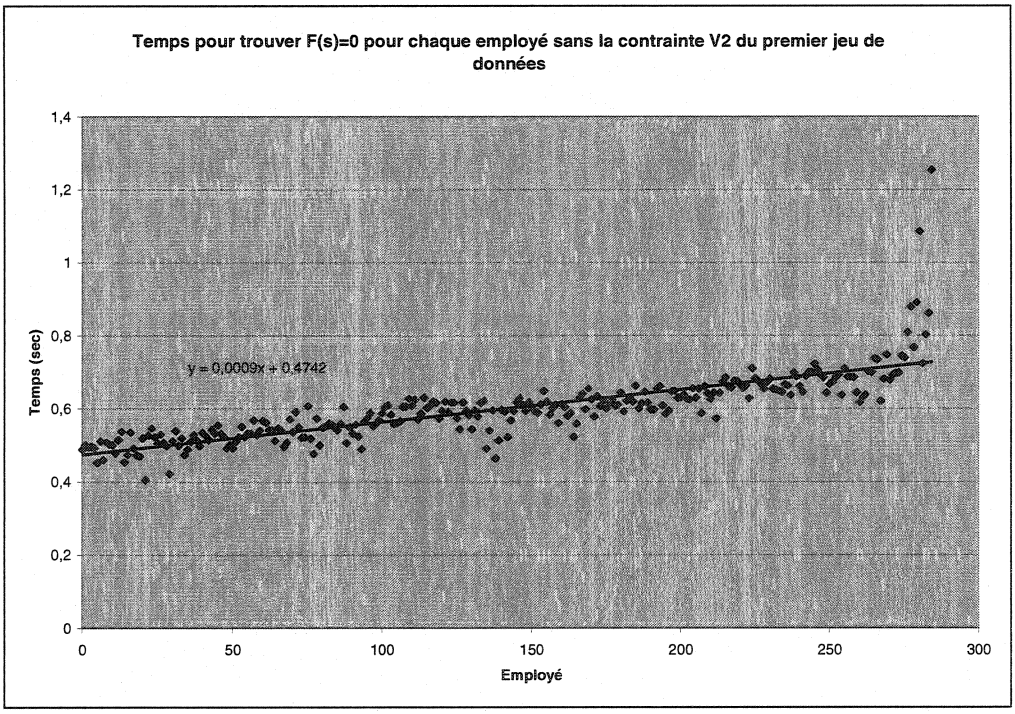


Figure 5.2 : Temps de calcul du premier jeu de données sans la contrainte V2

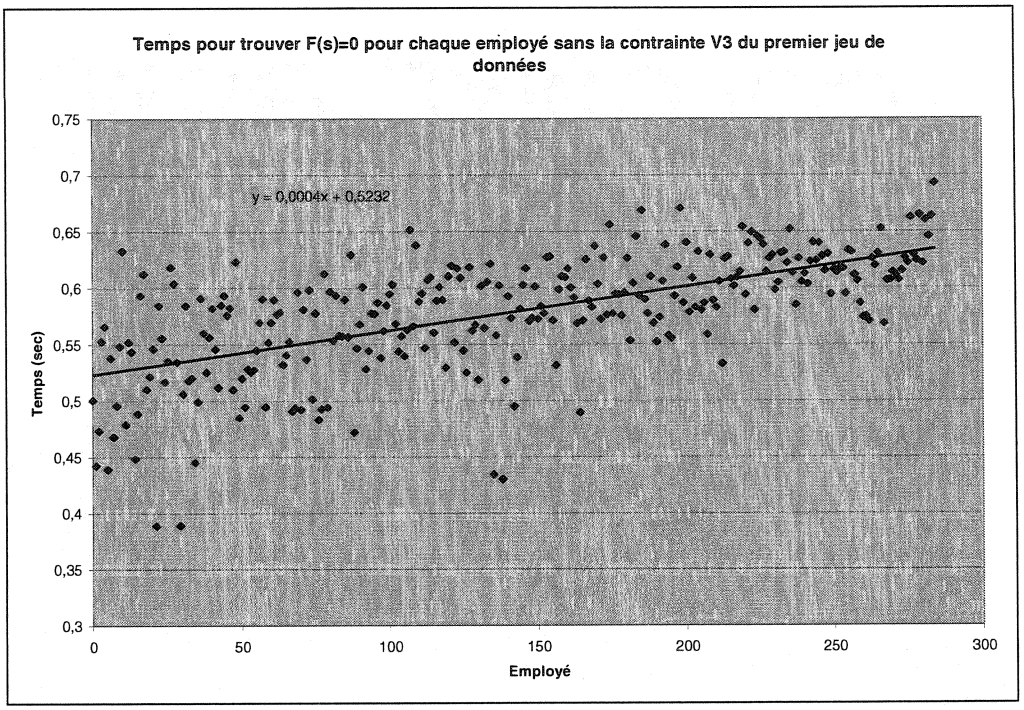


Figure 5.3 : Temps de calcul du premier jeu de données sans la contrainte V3

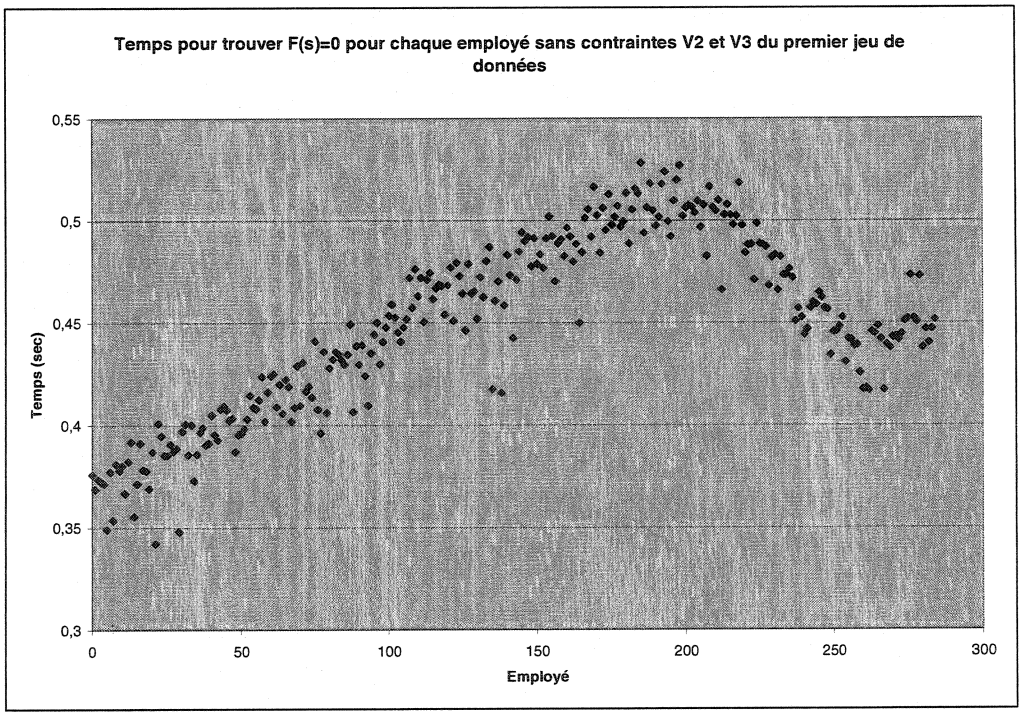
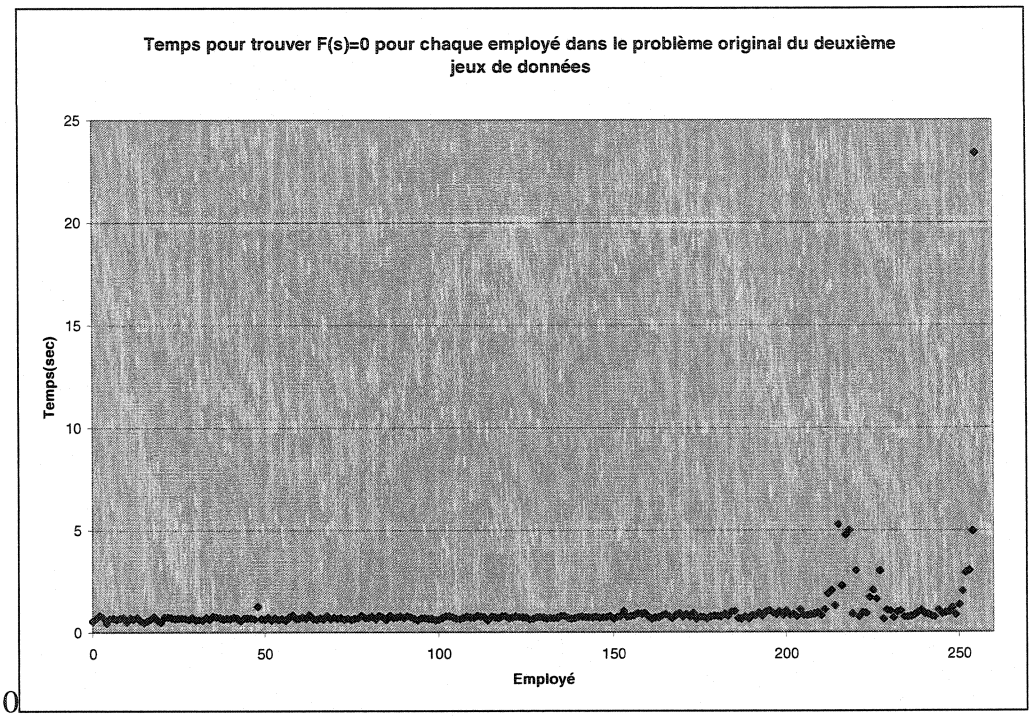


Figure 5.4 : Temps de calcul du premier jeu de données sans contraintes V2 et V3



0

Figure 5.5 : Temps de calcul du deuxième jeu de données problème original

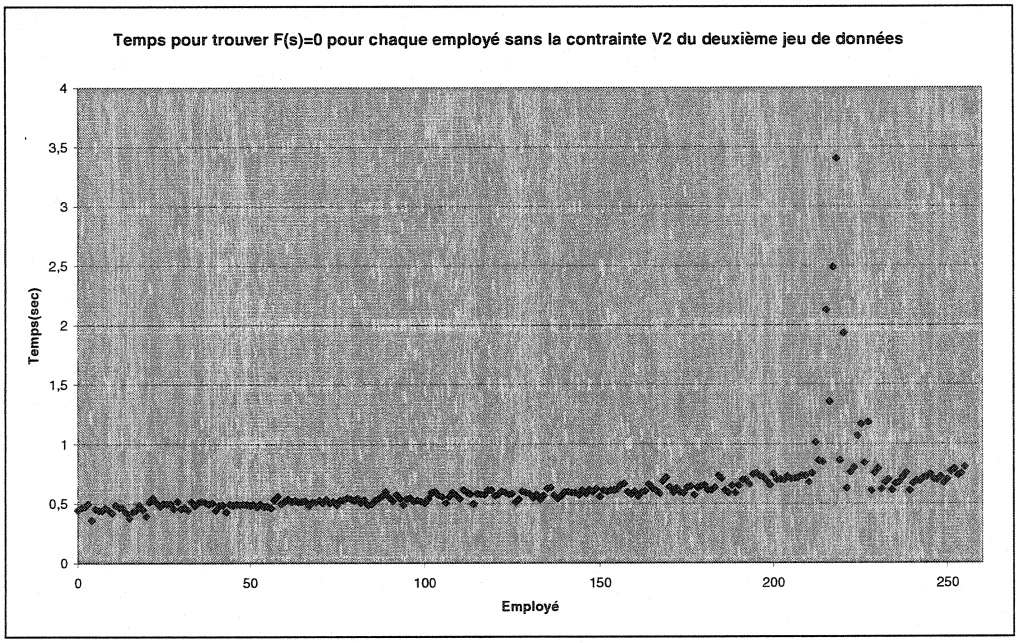


Figure 5.6 : Temps de calcul du deuxième jeu de données sans la contrainte V2

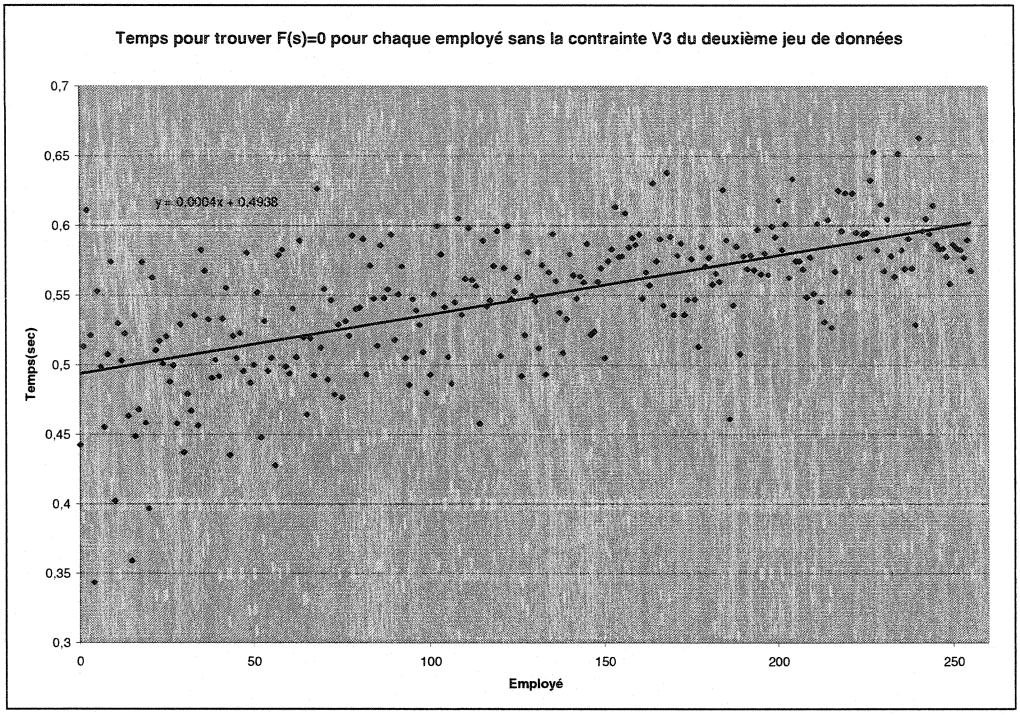


Figure 5.7 : Temps de calcul du deuxième jeu de données sans la contrainte V3

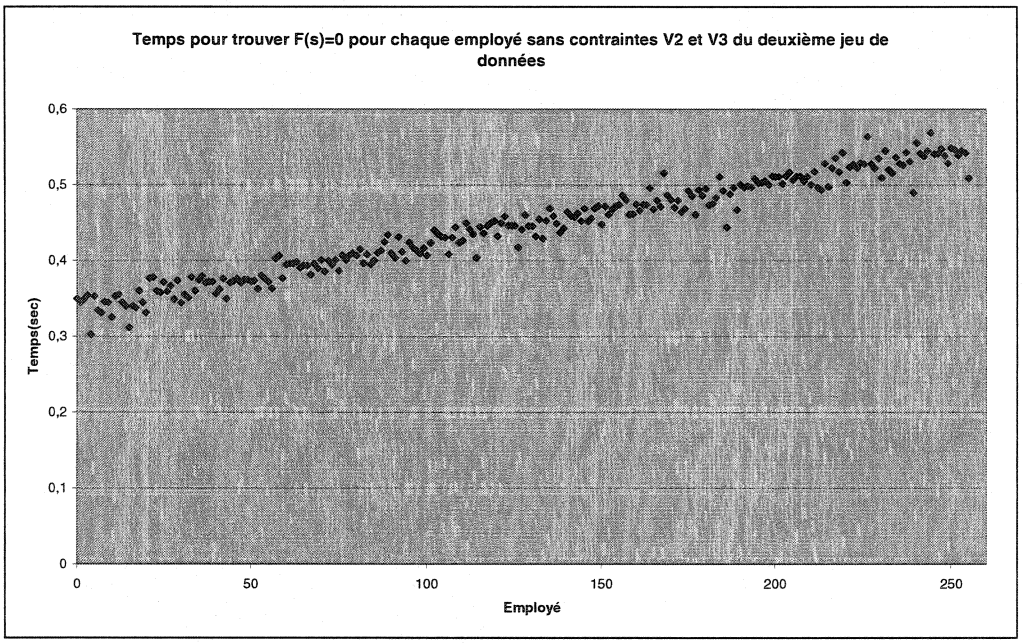


Figure 5.8 : Temps de calcul du deuxième jeu de données sans contraintes V2 et V3

Avec les huit figures précédentes et les temps moyens de calcul totaux suivants (tableau 5.4), quelques remarques peuvent être faites.

Tableau 5.4 : Temps moyens de calculs totaux

Jeux de données	Type de problème	Temps moyen de résolution (secondes)	Écart type
Jeu 1	Original	249,5	18,3
	Sans V2	186,6	4,9
	Sans V3	176,9	2,1
	Sans V2 et V3	138,8	14,1
Jeu 2	Original	284,5	46,7
	Sans V2	174,2	10,9
	Sans V3	153,7	1,0
	Sans V2 et V3	123,8	0,4

Premièrement, le temps de résolution baisse en enlevant les contraintes, ce qui est évident car le problème a moins de contraintes à satisfaire. Deuxièmement, il semble plus difficile de tenir compte des contraintes de type V3 que de celles de type V2, car la résolution est plus rapide en enlevant les contraintes de type V3 qu'en enlevant celles de type V2. Ceci est probablement dû au fait que chaque employé ajoute deux contraintes au problème, surpassant ainsi les contraintes créées par les tâches ayant des qualifications de langue. On peut cependant noter que dans le deuxième jeu de données, le programme a plus de difficulté à trouver une solution pour les employés 212 à 233. Cette difficulté n'est cependant plus présente dans les figures 5.7 et 5.8, ce qui porte à croire que pour ces employés la contrainte de type V3 était difficile à satisfaire. Afin de bien pouvoir comparer les deux jeux de données, seul les employés 0 à 211 seront pris en compte. Les quatre figures suivantes correspondent au deuxième jeu de données jusqu'à l'employé 211.

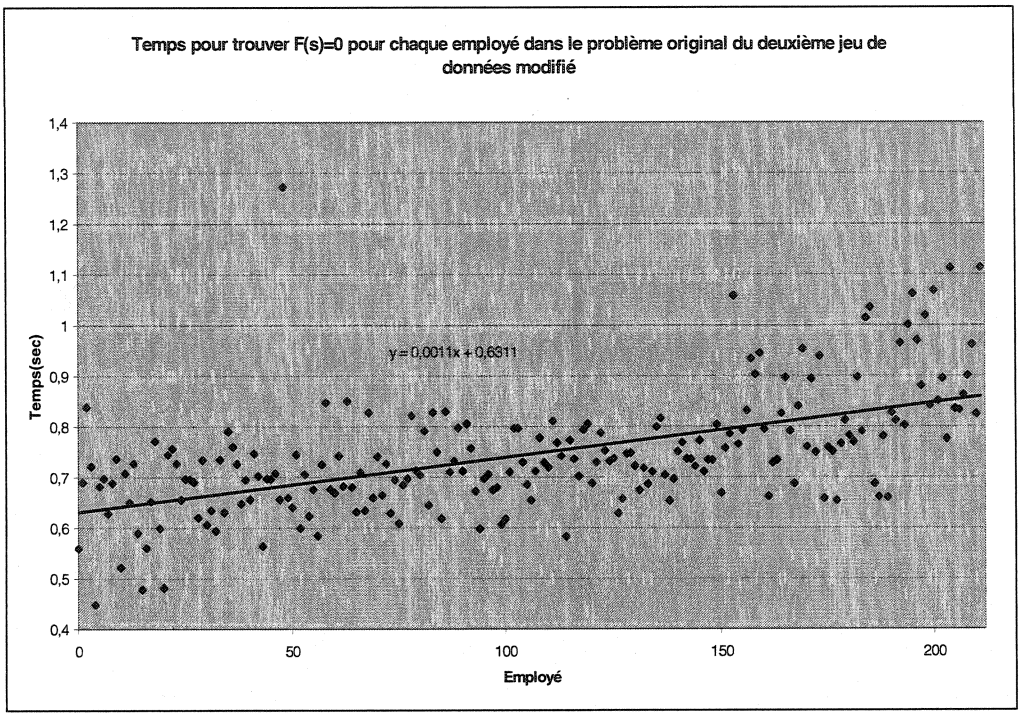


Figure 5.9 : Temps de calcul du deuxième jeu modifié problème original

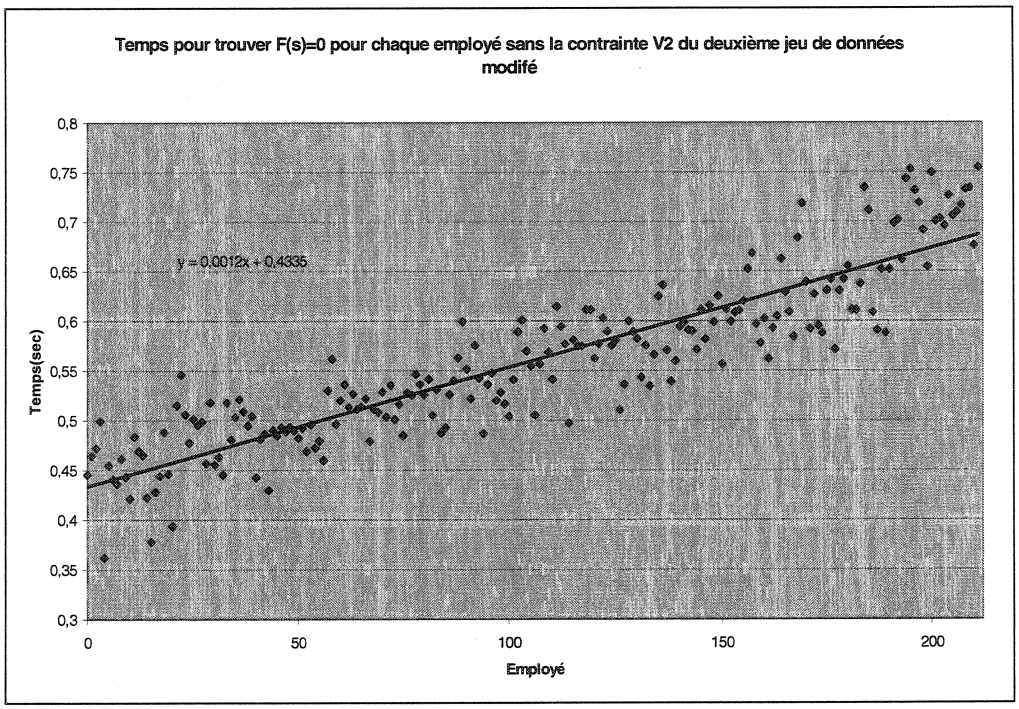


Figure 5.10 : Temps de calcul du deuxième jeu modifié sans la contrainteV2

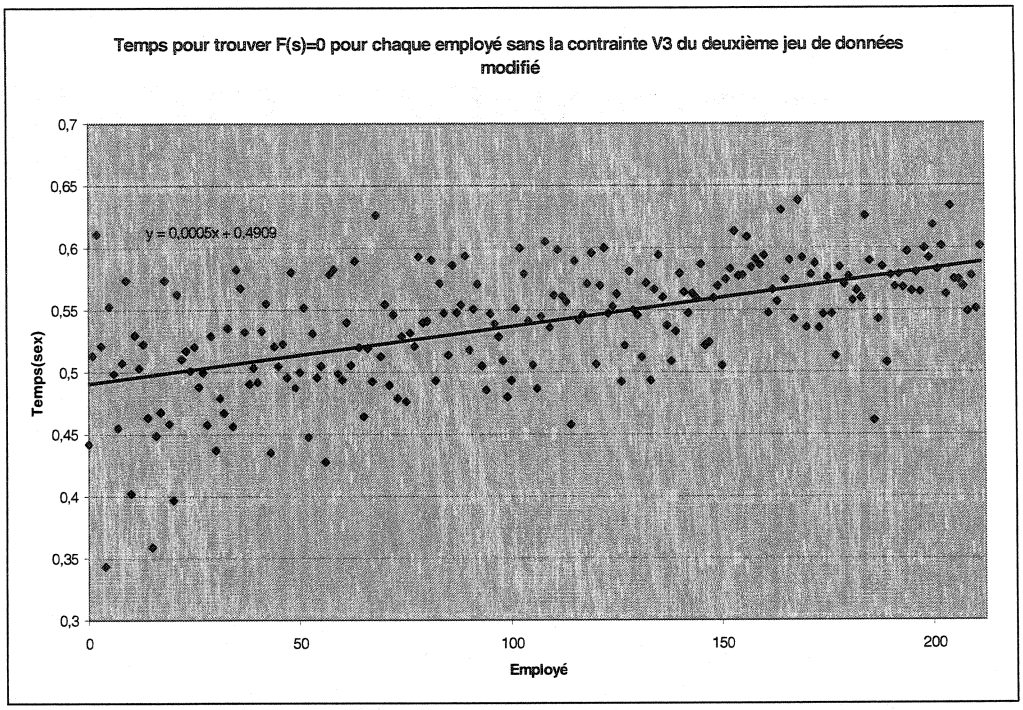


Figure 5.11 : Temps de calcul du deuxième jeu modifié sans la contrainte V3

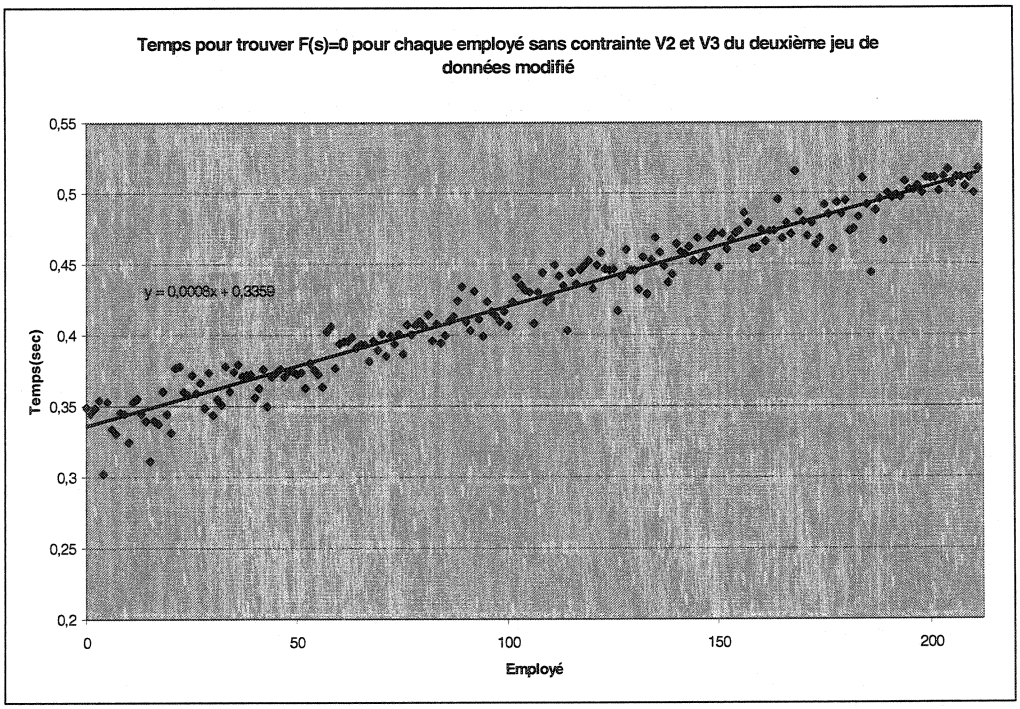


Figure 5.12 : Temps de calcul du deuxième jeu modifié sans contraintes V2 et V3

Comme le démontre la pente sur les graphiques, la résolution des employés devient de plus en plus difficile moins il reste d'employés dans le problème sauf dans la figure 5.4. On peut aussi remarquer que la résolution des employés semble être plus difficile pour le deuxième jeu que pour le premier car la pente des graphiques du deuxième jeu de données est plus élevée. La figure 5.13 incorpore les sept pentes pour mieux les visualiser. On ne peut pas tirer de conclusions sur ce qui pourrait se passer à droite des courbes présentées dans la figure 5.13 car le problème devient trop serré pour notre méthode et toute interprétation sur les tendance vers la droite sont trompeuses. En ce qui concerne la figure 5.4, les raisons entourant la forme de la courbe seront expliquées un peu plus loin.

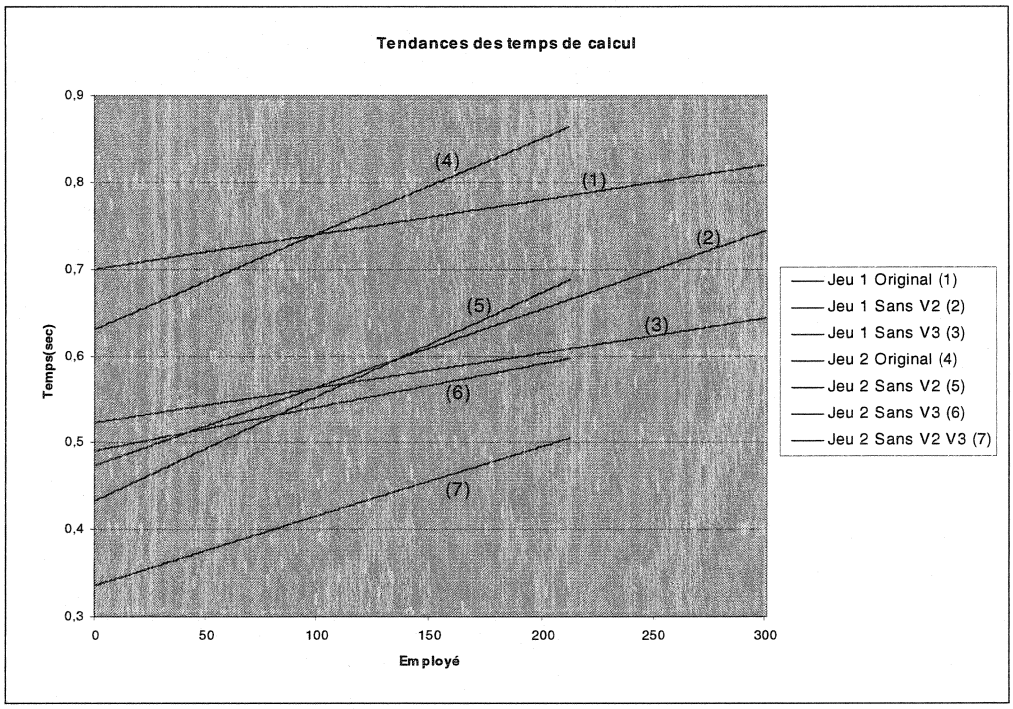


Figure 5.13 : Pentes des graphiques des figures 5.1 à 5.3 et 5.9 à 5.12

L'effet des contraintes peut aussi être vu dans la recherche d'une solution initiale des problèmes. Dans le tableau 5.5, le nombre d'itérations et le temps de calcul de la recherche d'une solution initiale des problèmes sont donnés. Ici aussi, les temps de calcul diminuent en enlevant les contraintes et suivent le même patron que la résolution entière du programme. Comme mentionné dans la section 3.6.1, l'initialisation du programme se fait à l'aide d'un algorithme glouton qui distribue les tâches aux employés. Suite à cette distribution, la recherche d'une solution initiale est faite. Le temps que prend cette recherche dépend donc de la qualité de la solution trouvée par l'algorithme glouton. Si on se fie au nombre d'itérations requis pour trouver une solution initiale, l'algorithme glouton semble avoir plus de difficultés à bien distribuer les tâches quant aux contraintes de type V3 qu'à celles de type V2. Sans les contraintes de type V2 et V3, l'algorithme glouton n'a pas de problème à distribuer les tâches sans conflits, ce qui évite totalement la phase de recherche d'une solution initiale.

Tableau 5.5 : Données sur la recherche d'une solution initiale

Jeux de données	Type de problème	Nombre d'itérations solution initiale	Temps de calcul solution initiale (sec.)
Jeu 1	Original	650	18,8
	Sans V2	390	4,6
	Sans V3	48	0,8
	Sans V2 et V3	0	0,0
Jeu 2	Original	896	20,8
	Sans V2	231	4,0
	Sans V3	53	1,0
	Sans V2 et V3	0	0,0

Une chose doit être prise en compte dans l'analyse et l'évaluation des données précédentes. Elle est reliée à la tendance au problème à devenir compliqué plus la résolution avance. Les pentes portent à croire que cet effet n'est pas très marqué mais il faut tenir compte du fait que moins il y a de tâches à assigner aux employés, plus l'algorithme tabou est rapide. Dans le cas présent, même si les calculs sont plus faciles, le temps total de résolution de chaque employé augmente. Ceci implique donc que la

résolution devient beaucoup plus ardue que le laissent croire les graphiques. Il est possible de remarquer ceci sur le graphique de la figure 5.4. La difficulté augmente plus le nombre d'employés résolus augmente. Après un certain temps cette difficulté est surpassée par la facilité à faire les calculs et les temps de résolution se mettent à baisser. Une fois que le nombre d'employés restants est restreint, les temps de résolution se remettent à augmenter puisque que peu d'employés sont disponibles pour faire les tâches restantes. Beaucoup d'échanges sont donc nécessaires avant de trouver un horaire légal pour tous les employés.

5.4 Amélioration des solutions actuelles

Un des endroits où la nouvelle méthode peut améliorer la solution est dans sa gestion des contraintes reliées aux rotations. La méthode des compteurs fait une distribution des qualifications sur les tâches. Par exemple, si une rotation requiert deux employés parlant allemand et un parlant polonais, la méthode des compteurs va demander deux employés parlant allemand et un parlant polonais, pour un minimum de trois employés. La nouvelle méthode quant à elle, ne distribue pas les qualifications sur les tâches mais considère l'ensemble des employés de la rotation. Si un employé parlant polonais et allemand peut être assigné à une tâche de la rotation, il va contribuer à remplir une demande en allemand et une demande en polonais. À plusieurs moments durant la résolution du problème, une telle situation se produit. Par exemple, avec la méthode des compteurs l'employé 195 du premier jeu de données, qui a la qualification allemand et polonais, doit faire une tâche requérant un polonais pour satisfaire une contrainte critique. La tâche en question fait partie d'une rotation requérant un polonais et un allemand. Plus tard, pour satisfaire une contrainte critique, l'employé 228 se doit de faire une tâche requérant un allemand de la même rotation. Par contre, avec la nouvelle manière de faire, l'employé 228 n'aurait pas à faire la tâche qui satisfait la contrainte critique car la partie allemande de la rotation est déjà satisfaite par l'employé 195. Il n'y aurait donc pas de contrainte critique allemande lorsqu'on traite l'employé

228. Si la tâche faisait partie des préférences de l'employé 228, il n'y a pas de problème, sinon la situation est bien fâcheuse car un meilleur horaire aurait probablement pu être trouvé pour l'employé 228.

5.5 Accélération des calculs

Bien du temps a été investi dans la recherche de moyens afin de diminuer les temps de calcul. Un de ces moyens est de permettre d'accepter le premier voisin améliorant (voir figure 3.12 ligne a.2). Accepter le premier voisin améliorant implique que si un changement voisin abaisse la valeur de la fonction objectif plus bas que la meilleure solution trouvée jusqu'à maintenant, la recherche du meilleur voisin est arrêtée et le changement est fait. Ceci permet de faire beaucoup moins de calculs de voisins, toutefois cette stratégie a pour inconvénients d'ignorer tout autre changement qui pourrait abaisser encore plus la valeur de la fonction objectif. Une résolution d'un problème modifié (considérant seulement les 28 premiers employés) a été faite avec une version de l'algorithme sans ce principe et le temps total a été de 201 secondes au lieu de 48. Accepter le premier voisin améliorant accélère donc les calculs d'un facteur d'environ 5.

Un autre bon moyen est l'utilisation des structures $\Delta V1$, $\Delta V2$ et $\Delta V3$. Le même problème considérant les 28 premiers employés a été résolus avec l'algorithme modifié plus tôt auquel on a enlevé en plus les structures ΔV. Le temps de résolution est passé à 920 secondes impliquant que les structures ΔV accélèrent les temps de calcul par un facteur d'environ 4. Cependant durant les tests, un effet non escompté des structures ΔV sur les temps de calcul a été remarqué. Si les structures ΔV ne sont pas utilisées durant l'affectation des employés aux tâches, les temps de calculs vont être plus rapides. En effet, en utilisant les structures ΔV, ces structures doivent être mises à jours une fois par itération. Pour faciliter l'explication, on supposera qu'il reste x tâches dans le problème

et y employés. Mettre à jours la structure $\Delta V1$ implique le calcul de la variation de V1 pour les deux employés impliqués pour les x tâches pour un total de $2x$ calculs. Pour ce qui est de la structure $\Delta V2$, il faudra calculer la variation de V2 pour toutes les tâches de la rotation impliquée, en moyenne deux par rotation, multiplié par les y employés restants ce qui donne environ $2y$ calculs. Pour la structure $\Delta V3$, sa mise à jours requiert $2x$ calculs, pour un grand total de $4x+2y$. Au début de la résolution d'un problème de 2000 tâches avec 300 employés ce nombre représente 8600 calculs. Si on considère les calculs nécessaires sans les structures ΔV, il faut faire un calcul de chaque type pour chaque voisin considéré. Avec en moyenne 894 voisins calculés, nous devons faire un peu moins de 3000 calculs par employé. Ceci implique donc que l'utilisation des structures ΔV n'est avantageuse que lorsqu'il y a beaucoup de voisins considérés, comme durant la recherche d'une solution initiale. Quand on fait cette recherche sans le principe d'acceptation du premier voisin améliorant, 25 millions de voisins sont considérés. C'est pourquoi l'utilisation des structures ΔV est si importante. La recherche d'une solution initiale prend 15 secondes dans notre version de l'algorithme et 892 pour une version sans structures ΔV et sans premier améliorant, ce qui donne un facteur de 60. La figure suivante montre l'effet des structures et de la stratégiedu premier améliorant sur les temps de calculs.

Tableau 5.6 : Temps de calculs sans les principes

	Temps pour recherche solution initiale (sec.)	Temps total (sec.)
Algorithme original	15	48
Sans premier améliorant	166	201
Sans Δ V	21	45
Sans premier améliorant et sans Δ V	892	920

Le tableau précédent montre que les structures ΔV devraient être utilisées seulement durant la recherche d'une solution initiale et que pour le reste de la résolution, la variation de la fonction objectif devrait être calculée pour chaque voisin considéré.

CHAPITRE 6 : CONCLUSION ET TRAVAUX FUTURS

Pour conclure, les objectifs visés par la recherche ont été atteints. Nous avons trouvé un moyen de modéliser le problème avec la coloration de graphes, et le nouvel algorithme permet de résoudre le problème.

Nous sommes aussi confiants que la solution trouvée par notre programme peut améliorer la satisfaction des employés face à leur horaire comme mentionné dans la section 5.4. Il reste cependant beaucoup de travail à faire pour que le nouveau programme créé puisse remplacer complètement la méthode des compteurs. Pour ce faire, les travaux suivants devront être faits.

Premièrement, il faudra trouver les contraintes critiques et les tâches qui les satisferont. Une étudiante se penche présentement sur le problème. Il faudra ensuite ajouter une partie au programme qui tient compte des préférences des employés afin de trouver leur meilleur horaire et d'assigner les bonnes tâches aux employés. Présentement, on tiens compte de cette partie en utilisant les meilleurs horaires trouvés par la méthode des compteurs qui elle tiens compte des préférences.

Les deux critères mentionnés plus tôt dans le texte devront aussi être évalués afin qu'ils soient pertinents. Le critère d'arrêt de l'algorithme pour un retour à la génération de colonnes, ainsi que le critère déterminant si des contraintes sont critiques dans le problème devront être étudiés.

Finalement, l'algorithme devra tenir compte de certaines contraintes que nous avons négligées dans l'élaboration de la version présente.

Plus tôt, dans le texte, nous avons mentionné qu'une insuffisance de crédits de vol peut se produire chez certains employés quand le meilleur horaire est assigné à

l'employé le plus senior. Pour régler ce problème, nous croyons que d'affecter dès le début le meilleur horaire à l'employé le plus senior règlera le problème. En assignant l'horaire dès le début, si l'assignation de ces tâches cause une insuffisance chez les employés résiduels, l'algorithme détectera ces insuffisances et l'horaire de l'employé le plus senior pourra être mis à jour en conséquence.

Nous avons aussi omis le fait que si une tâche est suivie d'un congé ou d'une autre préaffectation du genre, le repos suivant la tâche n'est plus obligatoire. Nous croyons qu'en créant deux catégories de tâches nous pourrons tenir compte de cette contrainte. Il suffira de vérifier quel type de tâche suivra la tâche à placer afin de savoir si le repos doit être ajouté ou non.

Finalement, il y a le concept de patron de repos dont nous n'avons pas tenu compte. Ce patron impose que l'horaire des employés suive un des patrons prédéfinis par la compagnie. Par exemple, un employé doit avoir 3 fois 3 jours de repos consécutifs dans le mois, ou bien il doit avoir 2 fois 5 jours consécutifs. Nous croyons qu'en créant une nouvelle catégorie de tâches qu'on pourrait situer entre les préaffectations et les tâches ordinaires, nous pourrons tenir compte de ce type de contrainte. Ces tâches représenteront les congés qu'un employé doit avoir durant le mois. Ces tâches n'auront pas de début ni de fin fixe, mais leurs durées seront données. Ces tâches seront groupées en ensembles respectant les différents patrons, et un de ces ensembles devra toujours faire partie des tâches de l'employé. Pendant la résolution, l'algorithme pourra choisir parmi les différents ensembles à sa guise afin de trouver une solution réalisable.

Malheureusement, le temps manquant, toutes ces hypothèses n'ont pu être mises à l'essai, mais nous sommes confiants qu'elles s'avèreront efficaces.

BIBLIOGRAPHIE

BIRÓ, M., HUJTER, M. et TUZA. Zs. (1992). Precoloring extension. I. Interval graphs. Discrete Math., 100(1-3), 267-279 .

BRÉLAZ, D. (1979). New methods to color vertices of a graph. Communications of the association for computing machinery, 22, 251-256.

CHASM, M., HERTZ, A. et De WERRA, D. (1987). Some experiments with simulated annealing for coloring graphs. European journal of operational research, 32, 260-266.

El IDRISSI, T (2002). Amélioration de la méthode des compteurs pour la construction des blocs mensuels personnalisés d'agents de bord. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

FLEURENT, C. et FERLAND, J.A. (1996). Genetic and hybrid algorithms for graph coloring. Annals of operations research, 63, 437-461.

GAMACHE, M., SOUMIS, F., VILLENEUVE, D., DESROSIERS, J. et GÉLINAS, E. (1998). The Preferential Bidding System at Air Canada. Transportation Science, 32(3), 246-255.

GERSHKOFF, I. (1989). Optimizing flight crew schedules. Interfaces, 19, 29-43.

GLOVER, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers and operations research, 13, 533-549.

HANSEN, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming. Congress on Numerical methods in combinatorial optimization, Capri, Italy.

HERTZ, A., COSTA, D. (1997). Ants can color graphs. Journal of the operational research society, 48, 295-305.

HERTZ, A., De WERRA, D. (1987). Using tabu search techniques for graph coloring. Computing, 39, 345-351.

JEANDROZ, P. (2000). Heuristique pour la construction de blocs mensuels personnalisés d'agents de bord. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

KARP, R.M. Reducibility among combinatorial problems. (1972). R.E. Miller, J.W. Thatcher, Complexity of Computer Computations, Plenum Press, New York, 85-103.

MITCHELSON, C. (2003). Nouvel algorithme de découpage pour la construction d'horaires mensuels personnalisés dans un contexte d'équité. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

ZUFFEREY, N. (2002). Heuristiques pour les problèmes de coloration des sommets d'un graphe et d'affectation de fréquences avec polarités. Thèse de doctorat, École Polytechnique fédérale de Lausanne, Suisse.